uc3m | Universidad Carlos III de Madrid

Grado Universitario en Ingeniería Informática
2018-2019

Trabajo Fin de Grado

"Uso de un robot humanoide para la transmisión de mensajes"

David Gutiérrez Fernández

Tutor

Ángel García Olaya

Leganés, junio de 2019

RESUMEN

Esta memoria describe el desarrollo de un sistema de intercambio de mensajes mediante la utilización del robot humanoide NAO y de Planificación Automática. Este sistema permite a un usuario emisor comunicarse con un usuario receptor mediante dos vías: enviando un correo al usuario receptor con el audio que contiene el mensaje adjunto si el mensaje a entregar es personal o entregando el mensaje el propio NAO mediante navegación guiada por pistas y reconocimiento facial del usuario receptor si el mensaje no es personal y por lo tanto no es importante que pueda ser escuchado por cualquier persona cercana al receptor.

Cuando se utilice la segunda vía, se permitirá al usuario receptor contestar al mensaje recibido, devolviéndose la respuesta por cualquiera de las dos vías mencionadas. Además, el usuario administrador del sistema gestionará el control de los usuarios autorizados para utilizar el sistema mediante un fichero que contendrá tanto los nombres de los usuarios como información que permite su reconocimiento facial al iniciar el sistema, asegurando así la correcta identidad del usuario.

Inicialmente, se ha descrito el estado de las tecnologías utilizadas en este trabajo y ejemplos reales de su aplicación. Después, para obtener el sistema descrito, se han definido los objetivos a alcanzar en el proyecto. Una vez definidos, se ha procedido al análisis del sistema mediante la utilización de los casos de uso y los requisitos. Con el análisis finalizado, se ha realizado el diseño de la arquitectura del sistema con su posterior implementación. Cuando la implementación del sistema se da por concluida, se realiza una batería de pruebas para comprobar su correcto funcionamiento.

Para la viabilidad del proyecto, se han tenido en cuenta, por un lado, el marco regulador para analizar las restricciones legales que pueda tener el proyecto y, por otro lado, el entorno socio-económico para realizar una planificación adecuada y un presupuesto esperado, con el correspondiente estudio del impacto socio-económico.

Finalmente, se comentan los objetivos alcanzados en el trabajo, las conclusiones obtenidas con su realización y las posibles futuras líneas de trabajo que permite este proyecto de cara a ampliar su alcance y complejidad.

Palabras clave

Sistema de intercambio de mensajes; Planificación Automática; PELEA; NAO; Python.

AGRADECIMIENTOS

Han pasado ya seis años desde que empecé esta aventura en la universidad. Durante este tiempo en la carrera ha habido momentos muy buenos y otros no tan buenos. Mi primer año fue un curso difícil de afrontar, con asignaturas complicadas que me sorprendieron y provocaron que mi experiencia universitaria en el primer curso de carrera no fuera nada agradable. Pero, sin embargo, aunque el resto de los años me haya ido bastante bien, considero mi primer año en la universidad el más importante no sólo en mi hábito de estudiante, sino lo que es en la vida misma, por muy negativo académicamente que fuera. Me enseñó que, si de verdad quieres algo, debes pelear por ello hasta el final, porque con voluntad y esfuerzo todo en la vida se puede conseguir.

Es posible que la exigencia en ciertos ámbitos nos haga sufrir bastante y nos obligue a dar lo mejor de nosotros mismos, pero es ahí donde reside la clave. Todo lo que vale la pena en esta vida hay que pelearlo y, aunque se pasen malos momentos, como durante este trabajo por ejemplo, uno se debe esforzar al máximo para, por un lado, mejorarse a sí mismo y, por otro lado, poder estar orgulloso de todo el trabajo realizado.

Quiero agradecer el apoyo de mis padres durante este camino, porque por mucho esfuerzo que yo haya puesto, sin ellos no habría sido posible conseguir nada. Además, quiero dar las gracias a compañeros como Alberto, Bea, Javi, Jesús, Luismi, Marcos, Paula y Víctor, porque juntos hemos podido avanzar, ayudándonos unos a otros en los momentos de dificultad. Sin olvidarme de Alba, ya que, aunque no haya coincido mucho durante la carrera con ella, sus consejos durante este trabajo han sido de gran ayuda.

Con respecto al personal de la universidad, quiero dar también las gracias a todos y cada uno de los profesores que he tenido en la carrera, porque en un grado mayor o menor, han influido en la persona que soy hoy en día. Afortunadamente, también he contado con el apoyo del grupo de Planificación y Aprendizaje del Departamento de Informática para todas las dudas que me surgían trabajando con el NAO y con la confianza que han depositado en mí a la hora de poder trabajar libremente con el robot. Por último, quería agradecer al tutor de este trabajo, Ángel García Olaya, toda la ayuda y atención que me ha prestado durante el desarrollo del proyecto, incluyendo sus consejos tanto en la implementación del trabajo como en la documentación de este, gracias de verdad.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1. INTRODUCCIÓN

Cuando una persona ve en la actualidad las características y variedad de los robots, puede preguntarse, ¿para qué necesitamos a los robots? ¿por qué existen los robots? La respuesta a ambas preguntas se basa en la necesidad del ser humano de crear máquinas inteligentes capaces de desarrollar actividades propias de los humanos.

Por tanto, una vez conseguida la creación de un robot, se le podrían asignar actividades que, por un lado, son tediosas y repetitivas o, por otro lado, suponen un riesgo físico para las personas que las realizan.

Sin embargo, la complejidad de un robot no sólo reside en su creación, sino que también está presente en el planteamiento de cómo otorgar inteligencia u autonomía a una máquina creada por el hombre. Si se consiguiera este objetivo, no sólo se tendría una máquina capaz de realizar tareas repetitivas, sino que también sería capaz de reaccionar ante una situación inesperada para poder seguir realizando su labor.

Dos conceptos que permiten la creación de un robot y la obtención de todas las características mencionadas son la Inteligencia Artificial y la Planificación Automática.

Por un lado, la Inteligencia Artificial es un campo de la informática que se encarga del estudio y creación de máquinas inteligentes capaces de realizar tareas que conllevan cierta complejidad.

Por otro lado, la Planificación Automática es una rama de la inteligencia artificial que le otorga a un robot la capacidad de, a partir de un problema a resolver, crear un plan de acciones que soluciona el problema y la capacidad de, ante una situación inesperada durante la ejecución del plan, realizar una replanificación, que consiste en la creación de un nuevo plan que se adapte a la situación actual en la que se encuentre el problema.

Conociendo el objetivo de los robots y las capacidades que le otorga a estos la Planificación Automática, el objetivo de este trabajo es utilizar al robot NAO mediante la Planificación Automática para la creación de un sistema intercambiador de mensajes. Este sistema permitirá a dos usuarios, mediante el reconocimiento facial, intercambiar mensajes entre ellos ya sea mediante el envío de un correo electrónico con el contenido del mensaje o mediante la entrega del propio NAO al receptor correspondiente.

1.1 Motivación del trabajo

La realización de este Trabajo de Fin de Grado se ha basado en dos tipos de motivación: la motivación del por qué es relevante realizarlo sobre el tema del que trata y la motivación personal del propio alumno por el tema.

Por un lado, realizar un trabajo sobre la creación de un intercambiador de mensajes utilizando al robot NAO es relevante porque supone la novedad de utilizar juntas tres características de bastante potencial:

- **Planificación Automática:** el problema del que trata este trabajo ha sido modelado utilizando la planificación automática y, por ello, el NAO, además de crear un plan de acciones para resolver el problema, podrá realizar la replanificación cuando se produzcan situaciones inesperadas.
- **Reconocimiento facial:** la utilización de esta característica se basará en el reconocimiento facial tanto del usuario emisor como del usuario receptor. El reconocimiento facial del usuario emisor estará orientado a identificarle y a comprobar que es un usuario autorizado mientras que el del usuario receptor estará orientado a la comprobación de que es el usuario al que el emisor quiere mandar el mensaje.
- **Navegación guiada:** esta característica otorgará al robot la capacidad de moverse por un entorno mediante la utilización de pistas con las que realizará una ruta que le llevará hasta la posición donde se encuentra el usuario receptor.

Por otro lado, la motivación personal por este tipo de trabajo, basado en la robótica, surgió por la idea de realizar un proyecto que fuera algo distinto a lo que se había realizado durante la carrera. Por ello, el tema de la programación web fue descartado rápidamente, dado la gran cantidad de trabajos presentes sobre este tema durante el grado.

Una vez teniendo claro que el tema del proyecto debía ser de algo que no se hubiera visto en exceso en la carrera, y teniendo en cuenta que el alumno realizó la rama de Sistemas de Información, surgió la idea de realizarlo sobre robótica.

A pesar de que es un tema que, evidentemente, pertenece a la rama de Computación, no se le dedica apenas tiempo en el grado y, de cara al futuro, puede tener importantes avances que doten de mayor autonomía e incluso consciencia propia a los robots.

Además, aparte de para obtener nuevos conocimientos, también ha sido una gran motivación trabajar con este tema de cara al ámbito laboral, ya que el tener conocimientos de robótica, aunque sean mínimos, siempre es interesante de cara a mejorar el currículum.

1.2 Estructura del documento

El documento se estructura en nueve capítulos. El primer capítulo es la introducción, que pretende describir el tema del trabajo, mostrar la motivación por el tema seleccionado y explicar la estructura de este documento.

El segundo capítulo describe el estado del arte, donde se realiza una investigación acerca del tema del trabajo y se muestran aplicaciones reales de las tres características explicadas en la introducción.

El tercer capítulo especifica los objetivos del trabajo, donde se enumeran aquellos objetivos que se pretenden conseguir con la creación del sistema que se va a desarrollar.

El cuarto capítulo muestra el análisis del sistema, donde se lleva a cabo una especificación detallada del sistema mediante la descripción de las características funcionales, la especificación del entorno operacional, la utilización de los casos de uso, la enumeración de los requisitos, que permiten captar las necesidades a resolver, y la revisión de que estas necesidades se cumplen mediante el empleo de una matriz de trazabilidad.

El quinto capítulo contiene el diseño y la implementación, donde se resuelve el problema planteado realizando la descripción de la arquitectura del sistema, la explicación de los pasos seguidos en su implementación y la enumeración de las alternativas de solución que han ido surgiendo durante la realización del trabajo.

El sexto capítulo presenta las pruebas del sistema, donde se muestra el entorno donde se han realizado las pruebas y se especifica un plan de pruebas que demuestra que todas las necesidades que se han planteado han sido resueltas de manera satisfactoria.

El séptimo capítulo contiene el marco regulador, donde se realiza un análisis de las restricciones legales que puede tener el proyecto, se muestran las licencias de las herramientas empleadas, se mencionan los estándares técnicos utilizados y se realiza un estudio de las cuestiones relacionadas con la propiedad intelectual de la idea.

El octavo capítulo describe el entorno socio-económico, donde se muestra la planificación que se ha realizado del proyecto, se especifica el presupuesto asociado a la realización del

trabajo y se realiza un estudio del impacto económico, social y ético que se espera que tendrá el proyecto.

El noveno capítulo contiene las conclusiones y trabajos futuros, donde se describen los objetivos que finalmente se han alcanzado con el Trabajo de Fin de Grado, se enumeran las conclusiones alcanzadas con su realización y se muestran posibles líneas de trabajo que permitan investigar y profundizar en este proyecto, de cara a ampliar su alcance y su complejidad.

2. ESTADO DEL ARTE

Como se ha mencionado en el capítulo anterior, con este trabajo se pretende utilizar al robot NAO para conseguir, mediante la Planificación Automática, construir un sistema que permita el intercambio de mensajes entre dos usuarios, utilizando durante su ejecución la navegación en un entorno y el reconocimiento facial.

La aplicación en este trabajo de la Planificación Automática viene reflejada en la utilización de PELEA, que es una arquitectura de código libre desarrollada por la Universidad Carlos III de Madrid, la Universidad Politécnica de Valencia y la Universidad de Granada.

A continuación, se procede a realizar una descripción de algunos de los robots humanoides más importantes en la historia de la robótica desde sus inicios hasta la actualidad, a especificar en qué consiste la Planificación Automática, a explicar los componentes y el funcionamiento de la arquitectura PELEA y, finalmente, a mostrar ejemplos actuales de navegación en un entorno y reconocimiento facial con robots.

2.1 Historia de la robótica

La robótica es la ciencia que tiene como objetivo el diseño, la construcción y la utilización de robots, que son máquinas programables capaces de realizar acciones que antes sólo podían realizar los humanos.

El término ***robot*** procede de la obra literaria *Rossum´s Universal Robots* de 1921, creada por el escritor checo *Karel Čapek*, originándose concretamente de la palabra *robota*, que en la obra significaba “labor forzada o esclavo” [1].

Con respecto al considerado como el primer robot de la historia, este es ***Elektro*** [2], un robot de dos metros de altura y ciento veinte kilogramos de peso. Su creador fue *Joseph Barnett* y su lugar de creación fue la fábrica de *Westinghouse* en *Mansfield, Inglaterra*, en 1937:

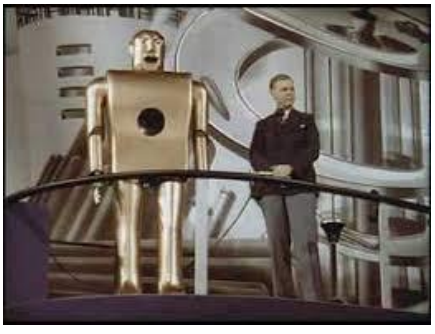


Ilustración 1: Robot Elektro

Este robot era capaz de hablar (con un vocabulario de setecientas palabras pregrabadas) y tenía ojos fotoeléctricos que le permitían distinguir entre la luz roja y la luz verde. Además del habla, también tenía como capacidades mover la cabeza y los brazos, fumar cigarrillos e inflar globos. Adquirió un gran protagonismo en la Feria Mundial de Nueva York de 1939, donde supuso una gran revolución tecnológica para la época.

Como se puede observar, Elektro se trataba de un robot humanoide con unas capacidades que hoy en día se consideran muy básicas en el mundo de la robótica, pero que en el momento de la exposición supuso un gran asombro por ser para mucha gente la primera vez que veía a un robot y le veía realizar acciones propias de un ser humano.

Aunque Elektro de forma visual se trataba de un robot humanoide, lo cierto es que su tren inferior (piernas) no tenía funcionalidad, sino que suponía una parte estética del robot que lo hacía más humano.

El hecho de conseguir que un robot tenga la coordinación necesaria para poder andar no era nada sencillo. Una aproximación interesante a la consecución de dicho objetivo se dio en el año 1986, con el modelo ***E0*** de la compañía *Honda* [3]:

Ilustración 2: Robot E0

Dicho modelo sólo disponía de piernas y era un modelo centrado en conseguir que un robot fuera capaz de realizar sus primeros pasos. Este robot fue capaz de conseguirlo, aunque sólo pudiera andar en línea recta y pasaran cinco segundos entre cada paso.

Tras la realización de distintos modelos, *Honda* creó el modelo ***E6*** [4] en 1993, que, aunque seguía siendo sólo la parte inferior de un robot humanoide, era capaz de mantener el equilibrio, lo que le permitía subir o bajar escaleras y esquivar obstáculos:

Ilustración 3: Robot E6

También en 1993, una vez que consiguió un tren inferior de un robot con bastante funcionalidad, *Honda* sacó el modelo ***P1*** [5], que supuso su primer robot humanoide completo. Dicho robot ya podía encender y apagar interruptores, agarrar los pomos de las puertas y llevar objetos:

Ilustración 4: Robot P1

Unos modelos más tarde, en el año 2000, *Honda* sacaría un robot revolucionario para la época de la que se trataba, el robot ***Asimo***. Tenía una altura de ciento veinte centímetros y un

peso de cincuenta kilos. Teniendo en cuenta su altura, podía alcanzar a apagar y encender interruptores, coger objetos de las mesas y verse cara a cara con una persona sentada en una silla [6]. Sus dos funcionalidades más interesantes eran su capacidad de andar, pudiendo alcanzar los mil seiscientos metros por hora, y la naturalidad de los gestos que realizaba:

Ilustración 5: Robot Asimo

En el año 2004, salió el mercado la primera versión del robot utilizado en este trabajo, el robot ***NAO***, de la compañía *Aldebaran Robotics*:

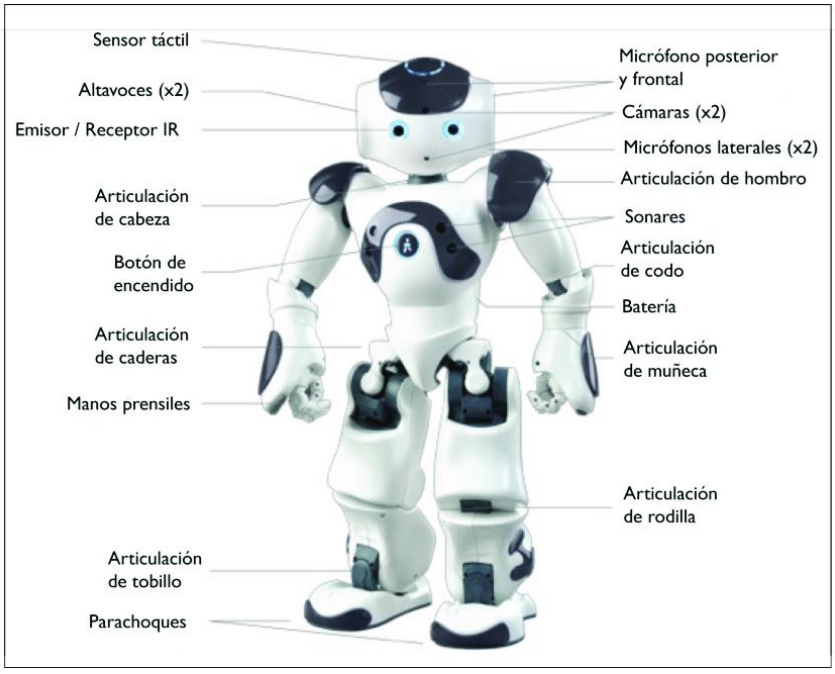


Ilustración 6: Robot NAO

Como se puede observar en la imagen [7], NAO posee una gran movilidad en sus articulaciones que le permiten realizar un gran número de movimientos (levantarse, tumbarse o mantenerse de pie con una sola pierna), además de tener importantes herramientas como

sonares, cámaras, altavoces, micrófonos, sensores infrarrojos y táctiles. Todas estas herramientas le permiten hablar, grabar y reproducir vídeos y audios, detectar y rastrear objetos, o reconocer personas.

En el año 2014, un reconocido ingeniero japonés llamado *Hiroshi Ishiguro*, creo el robot ***Geminoid HI-4***, que tiene como peculiaridad que es una copia exacta de su creador:

Ilustración 7: Robot Geminoid HI-4

Este robot, aunque sólo pueda utilizar de su cuerpo la cabeza, es capaz de impartir clases y conferencias, conversar con humanos, realizar gestos faciales propios de las personas y mover los ojos [8].

El último robot del que se va a hablar en este apartado es el robot ***Atlas***, de la compañía *Boston Dynamics*, que tiene todas sus características centradas en el movimiento. Mide ciento setenta y cinco centímetros y pesa ochenta y dos kilogramos. Tiene como habilidades evitar obstáculos, evaluar el terreno y orientarse [9]. Con ello, se puede desplazar por cualquier terreno por muy irregular que sea, puede abrir puertas y puede coger objetos del suelo [10]:

Ilustración 8: Robot Atlas

Su aplicación futura será realizar misiones que presentan cierto peligro para los humanos, como la actuación ante un desastre natural o un incendio.

2.2 Planificación Automática

La Planificación Automática es una disciplina de la Inteligencia Artificial cuyo objetivo es la producción de planes que permitan llegar a unas determinadas metas. Tiene su origen en el control de robots autónomos y este campo sigue constituyendo una de sus aplicaciones principales.

En este contexto, un plan es una secuencia ordenada de acciones que se deben realizar para llegar a un objetivo o meta. Por tanto, la Planificación Automática permite a un robot, a partir de un problema a resolver, obtener y ejecutar un plan de acciones que le permitirán llegar a su objetivo.

Un interesante ejemplo de uso de la Planificación Automática es en las misiones espaciales de exploración en Marte [11], concretamente en la utilización de un ***rover***, que es un vehículo de exploración.

Inicialmente, se tiene una posición inicial de la que parte el *rover* y una posición final que debe alcanzar. La siguiente imagen muestra la ruta que debe seguir el *rover* para alcanzar la posición final sin tener en cuenta los obstáculos presentes. Los iconos cuadrados son los puntos por los que debe atravesar el *rover*, denominados *waypoints*, y son determinados por un usuario. En cambio, los puntos circulares representan los lugares donde se llevarán a cabo actividades científicas:

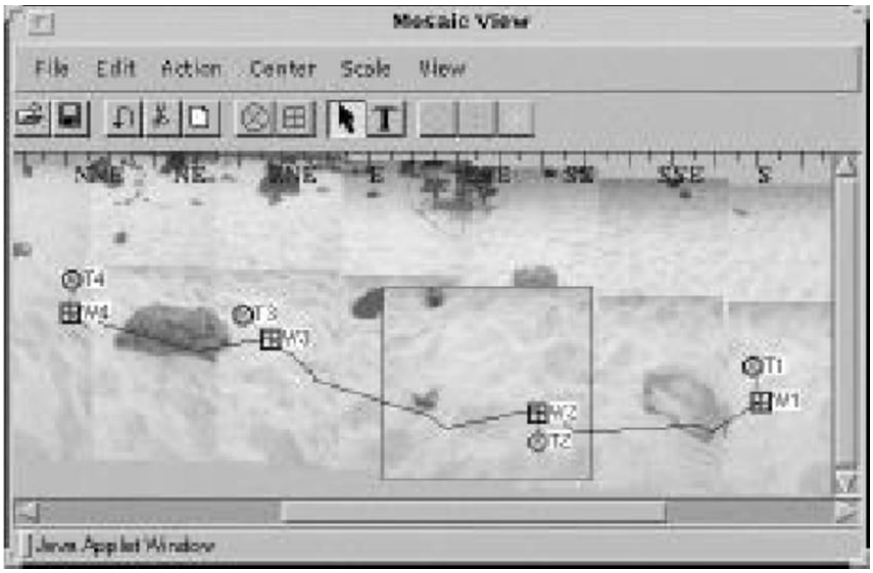


Ilustración 9: Ruta del rover sin detección de obstáculos

Una vez que los obstáculos son tenidos en cuenta, se realiza una replanificación del camino a recorrer esquivando los obstáculos presentes en la ruta de la imagen anterior:

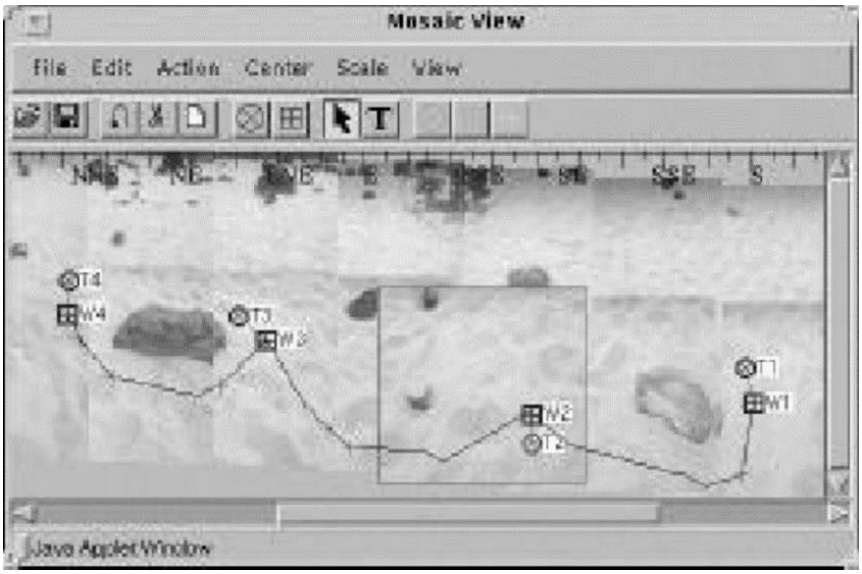


Ilustración 10: Ruta del rover con detección de obstáculos

La aplicación de la Planificación Automática generalmente suele ser realizada mediante planificadores automáticos, que son programas que incorporan algoritmos específicos de la Planificación Automática.

Una vez visto el concepto de Planificación Automática, es necesario para comprender su funcionamiento ver la manera en la que se puede expresar un problema que se quiera resolver utilizando dicha disciplina. Para ello, lo que se utilizan son los lenguajes de planificación, que permiten expresar todas las condiciones de un problema para su posterior resolución. Aunque hay bastantes lenguajes de planificación, en este apartado se va a describir PDDL, dado que se trata de un intento de estandarización de los lenguajes de planificación y es el que se va a utilizar en este trabajo.

PDDL (*Planning Domain Definition Language*) es un lenguaje de planificación automática desarrollado por *Drew McDermott*, e inspirado en otros lenguajes predecesores como STRIPS (*Stanford Research Institute Problem Solver*) y ADL (*Action Description Language*) [12].

Para su utilización, se divide la tarea a resolver en un dominio y en un problema. De cara a explicar cada una de las secciones que contienen el dominio y el problema se presenta un problema real como ejemplo en el que un robot, por ejemplo, el robot NAO, se encuentra en

una posición cero y debe conseguir llegar a la posición seis, pasando antes por la posición uno y por la posición tres, en ese orden:

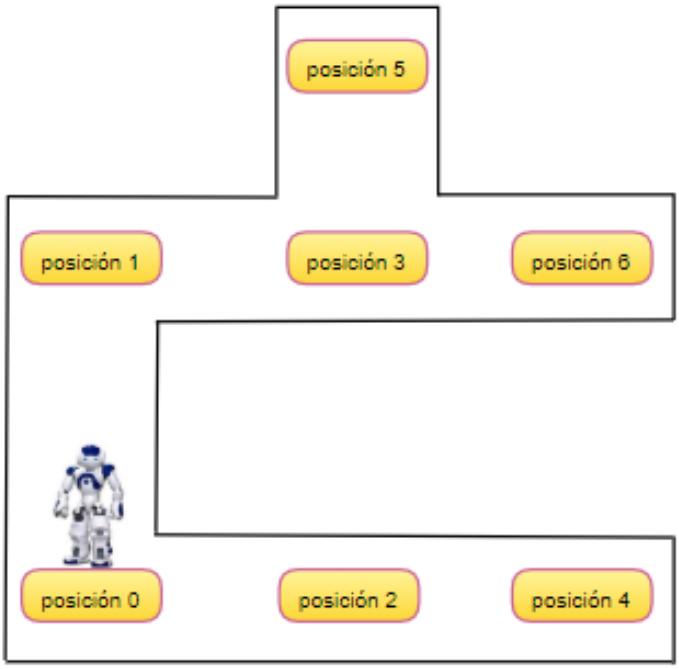


Ilustración 11: Ejemplo de PDDL

Teniendo este problema, se podría modelar el siguiente dominio:

```
(define (domain Example)

    (:types
        robot
        posicion
    )

    (:predicates
        (available ?r - robot)
        (at ?r - robot ?x - posicion)
        (connected ?x - posicion ?y - posicion)
    )

    (:action move
        :parameters (?r - robot ?x - posicion ?y - posicion)
        :precondition (and (available ?r)(at ?r ?x)(connected ?x ?y))
        :effect (and (not(at ?r ?x))(at ?r ?y))
    )

)
```

Ilustración 12: Dominio PDDL de ejemplo

Por tanto, en un dominio se tienen las siguientes secciones:

- **Define:** se utiliza esta sección para poner la palabra *domain* seguido del nombre que tendrá el dominio.
- **Types:** se utiliza para representar los tipos de objeto que se van a utilizar. El término objeto en este ámbito se utiliza para referirse a un tipo de entidad. En el ejemplo, como se trabaja con un robot y con posiciones para alcanzar, hay que añadir dos tipos: un tipo *robot* y un tipo *posicion*.
- **Predicates:** se utiliza para representar los predicados. Un predicado es una declaración en la que se muestra el estado de los objetos o su relación entre ellos. En el ejemplo se tienen los siguientes predicados:
  - *(available ?r - robot)*: el robot ***r*** se encuentra disponible.
  - *(at ?r - robot ?x - posicion)*: el robot ***r*** se encuentra en la posición ***x***.
  - *(connected ?x – posicion ?y - posicion)*: las posiciones ***x*** e ***y*** están conectadas.
- **Action:** se utiliza para representar cada una de las acciones disponibles. En el ejemplo se tiene una única acción *move*, que como toda acción se compone de las siguientes secciones:
  - **Parameters:** son los parámetros (objetos) que necesita la acción. En el caso de la acción *move*, se le pasa un objeto *robot* y dos objetos *posicion*.
  - **Precondition:** son las precondiciones de la acción. Las precondiciones son aquellas condiciones que se tienen que cumplir para que la acción pueda realizarse. En el caso de la acción *move*, como precondiciones se tiene que el robot ***r*** esté disponible, que se encuentre en la posición ***x*** y que las posiciones ***x*** e ***y*** se encuentren conectadas. En caso de que se cumplan todas estas condiciones, la acción *move* podrá realizarse.
  - **Effect:** son las postcondiciones de la acción. Las postcondiciones son las consecuencias que tiene la realización de la acción. En el caso de la acción *move*, el robot ***r*** ya no se encontrará en la posición ***x***, sino que se encontrará en la posición ***y*** al haberse desplazado.

Para el dominio anterior, se modelaría el siguiente problema:

```
(define (problem ExampleProblem)

    (:domain
        Example
    )

    (:objects
        robot0 - robot
        posicion0 - posicion
        posicion1 - posicion
        posicion2 - posicion
        posicion3 - posicion
        posicion4 - posicion
        posicion5 - posicion
        posicion6 - posicion
    )

    (:init
        (available robot0)
        (at robot0 posicion0)
        (connected posicion0 posicion2)
        (connected posicion2 posicion0)
        (connected posicion2 posicion4)
        (connected posicion4 posicion2)
        (connected posicion0 posicion1)
        (connected posicion1 posicion0)
        (connected posicion1 posicion3)
        (connected posicion3 posicion1)
        (connected posicion3 posicion5)
        (connected posicion5 posicion3)
        (connected posicion3 posicion6)
        (connected posicion6 posicion3)
    )

    (:goal
        (at robot0 posicion6)
    )

)
```

Ilustración 13: Problema PDDL de ejemplo

Por tanto, en un problema se tienen las siguientes secciones:

- **Define:** se utiliza esta sección para poner la palabra *problem* seguido del nombre que tendrá el problema.
- **Domain:** se utiliza esta sección para poner el nombre del dominio al que hace referencia este problema.
- **Objects:** se utiliza para representar a los objetos y los tipos a los que pertenecen. En el ejemplo, al haber un robot y siete posiciones se declararán ocho objetos, uno de tipo *robot* y siete de tipo *posicion*.
- **Init:** se utiliza para representar los predicados que caracterizan al estado inicial del problema. En el ejemplo se ponen los predicados necesarios para indicar que el robot está disponible, para especificar que se encuentra en la posición cero y para realizar

las conexiones de aquellas posiciones que se encuentren conectadas y por tanto se pueda ir desde una a la otra.

- **Goal:** se utiliza para indicar el estado objetivo o meta que se debe alcanzar para que el problema se solucione. En el ejemplo, el problema se habrá solucionado cuando el robot se encuentre en la posición seis.

Una vez que ya se ha definido el dominio y el problema del ejemplo, se pasan ambas partes a un planificador automático [13], que obtiene el plan con el que el problema quedaría solucionado:

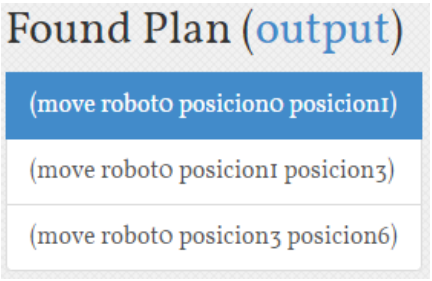


Ilustración 14: Resolución de planificador automático

Al tratarse de un problema sencillo, la solución era bastante lógica (el robot va primero a la posicion uno, luego a la tres y finalmente a la seis), pero el potencial que tiene la Planificación Automática permite resolver de esta misma forma problemas con bastante más complejidad.

En caso de que se quiera obtener más información de PDDL, se puede consultar el documento referenciado [14].

2.3 Arquitectura PELEA

Una vez que ya han sido explicados y comprendidos los conceptos de Planificación Automática y PDDL, es posible realizar una descripción acerca de PELEA.

PELEA (*Planning*, *Execution* and *Learning Architecture*) es una arquitectura de código libre que permite la planificación, ejecución, monitorización, replanificación y aprendizaje de tareas orientado al control de robots.

Dicha arquitectura trabaja en PDDL, partiendo inicialmente de un dominio y un problema a resolver. Durante su ejecución, PELEA presenta dos niveles diferentes:

- **Alto nivel:** relacionado con las acciones que genera el planificador automático.
- **Bajo nivel:** relacionado con las acciones que realiza el robot.

La versión más básica, que es la utilizada en este trabajo, consta de tres módulos que colaboran entre ellos, como se puede ver en la siguiente imagen:

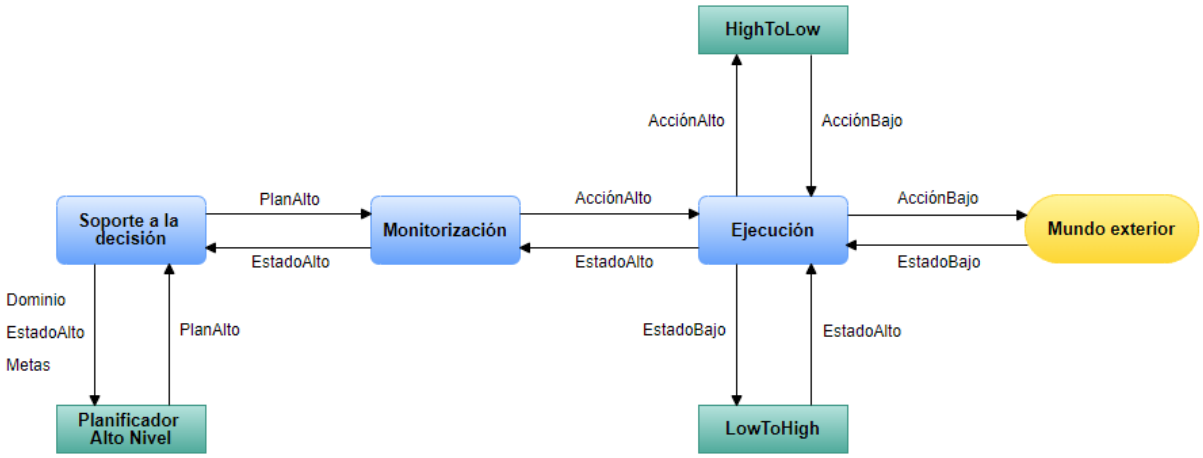


Ilustración 15: Funcionamiento de PELEA

La función que realiza cada uno de los módulos es la siguiente:

- **Soporte a la decisión:** este módulo recibe inicialmente un dominio y un problema del módulo de *Monitorización*. Entonces, manda ambos a un componente que contiene un *Planificador de Alto Nivel*, que le devuelve un plan con las acciones a realizar para llegar a la meta. Una vez que ya tiene el plan, este es mandado al módulo de *Monitorización*. En el caso ideal de que las acciones del plan obtenido se realizaran correctamente sin ningún estado inesperado, este módulo tras haber generado el plan inicial no volvería a intervenir. Pero lo más habitual, es que durante la ejecución del plan surjan estados inesperados que provoquen la replanificación. En caso de que esto ocurra, este módulo recibirá del módulo de *Monitorización* el estado real del problema a alto nivel para que, a partir de él, junto con el dominio del problema y las metas que se deben alcanzar, llame de nuevo al componente con el *Planificador de Alto Nivel* para obtener un nuevo plan para llegar a las metas.
- **Monitorización:** se trata del módulo que actúa de controlador. Al iniciarse la ejecución, manda el dominio y el problema a resolver al módulo de *Soporte a la decisión*, recibiendo de él un plan de acciones de alto nivel a realizar. Entonces, se encarga de mandar cada acción de alto nivel al módulo de *Ejecución*. Una vez mandada la acción de alto nivel, lo que recibe del módulo de *Ejecución* es el estado real en el que se encuentra el problema a alto nivel. En este punto, si el estado recibido es el esperado, se manda la siguiente acción de alto nivel del plan al módulo de *Ejecución*. Sin embargo, en caso de que el estado recibido no se corresponda con el

esperado, será necesario realizar una replanificación mandando al módulo de *Soporte a la decisión* el estado real del problema a alto nivel, para que a partir de él realice un nuevo plan para llegar a las metas.

- **Ejecución:** este módulo recibe del módulo de *Monitorización* una acción de alto nivel del plan. Entonces, utilizando el componente *HighToLow*, se encarga de traducir la acción de alto nivel en una o varias acciones de bajo nivel (en este trabajo, cada acción de alto nivel se corresponde con una única acción de bajo nivel). Una vez que tiene la acción o acciones de bajo nivel, estas deberían ser mandadas al robot para que las realizara. Cuando el robot realizara las acciones, este módulo debería recibir el estado real a bajo nivel en el que se encuentra el problema. Entonces, utilizando el componente *LowToHigh*, traduciría el estado real del problema de bajo nivel a alto nivel. Una vez obtenido el estado de alto nivel, este sería mandado al módulo de *Monitorización*.

En caso de que se quiera obtener más información sobre PELEA, se puede acudir al siguiente documento referenciado [15].

2.4 Trabajos relacionados

Una vez que se explicó de que trata la Planificación Automática y se mostró un ejemplo concreto de su aplicación, se presentan una serie de ejemplos que ilustran el estado de avance se encuentran las otras dos bases del trabajo: navegación en un entorno y reconocimiento facial.

Con respecto a la navegación en un entorno, se trata de uno de los temas más complejos a los que se enfrenta la robótica, puesto que en muchas ocasiones el robot se encontrará en un entorno no controlado en el que deberá realizar un recorrido por él. Que el entorno no esté controlado implica muchos posibles problemas, como son los siguientes:

- Suelo irregular que hace que el robot no pueda continuar andando y haga que se caiga al suelo.
- Condiciones meteorológicas adversas que pueden provocar el deterioro de ciertos robots.
- Mapeado de la zona que permita al robot conocer el recorrido que debe realizar para alcanzar la posición final.

- Control de la posición actual del robot, ya que será necesario que el mismo robot sea capaz de ubicarse en el mapa de la ruta para poder conocer a qué distancia se encuentra de su objetivo.
- Aparición de posibles obstáculos durante el recorrido que impliquen realizar pequeñas desviaciones con el objetivo de esquivarlos.

A pesar de todas estas dificultades, hoy en día existen robots capaces de realizar sus respectivas rutas gracias al empleo de técnicas SLAM (*Simultaneous Localization And Mapping*), que son técnicas de navegación que permiten el mapeo de entornos desconocidos.

En primer lugar, el robot ***FedEx SameDay Bot***, de la compañía *FedEx*, es un robot capaz de realizar entregas de corto alcance mediante el uso de sensores LIDAR y de algoritmos de planificación de rutas [16], pudiendo detectar obstáculos y recorrer superficies irregulares [17]:

Ilustración 16: Robot FedEx SameDay Bot

En segundo lugar, el robot ***Roomba 981***, de la compañía *iRobot*, es un robot aspirador capaz de realizar la limpieza de suelos de una planta mediante el empleo de la tecnología SLAM para no repetir la limpieza en zonas ya visitadas y controlar su posición y el empleo de la tecnología *iAdapt* [18], que permite esquivar obstáculos y evitar desniveles [19]:

Ilustración 17: Robot Roomba 981

En tercer y último lugar, se encuentran los ***Vehículos de Guiado Automático*** (Automated Guided Vehicles), que se tratan de vehículos no tripulados para el transporte de materiales en las fábricas [20]. El guiado de dichos vehículos puede ser implementado de distintas formas:

- **Filoguiado:** el vehículo se desplaza utilizando un hilo conductor instalado en el suelo.
- **Optoguiado:** el vehículo se desplaza utilizando tiras de espejo que le indican los movimientos que debe realizar.
- **Visión artificial:** el vehículo reconoce las tiras de espejo catadióptrico que le indican el camino mediante el uso de visión artificial.
- **Guiado láser:** el vehículo utiliza una unidad láser giratorio y espejos catadióptricos para la realización de su ruta [21].
- **Mapeado 2D-3D:** el vehículo posee un mapa del entorno donde se mueve que le permite conocer la posición de los posibles obstáculos y su posición final.

Con respecto al reconocimiento facial, se trata de un tema que en el último año ha tenido increíbles alcances. El reconocimiento facial supone un método de seguridad robusto siempre que dicho reconocimiento sea eficaz y tenga una alta tasa de acierto. Este reconocimiento a veces se complica porque, por ejemplo, el ladrón de una tienda tiene el rostro mirando a un punto distinto al que tiene la cámara y, por tanto, no se dispone de una correcta visión del rostro para realizar el reconocimiento. Además, siempre existe el peligro de que el delincuente disponga de una foto de una persona autorizada para autenticarse como esta y saltarse así las medidas de seguridad.

Un ejemplo de reconocimiento es el implementado por el robot ***Pepper***, propiedad de la compañía *Softbank Robotics*, al igual que el robot NAO:

Ilustración 18: Robot Pepper

Este robot es capaz de realizar el reconocimiento de personas que se acercan a él, pudiendo además de poder decir su nombre orientar a las personas sobre el lugar y la hora de sus reuniones o realizar llamadas [22]. En el caso de este robot, los avances en el reconocimiento facial van en aumento gracias a las actualizaciones que le realiza la empresa *Ever AI* [23], las cuales dotan al Pepper de las siguientes características en el reconocimiento [24]:

- **Detección de vida:** impidiendo con ello el uso de fotografías para suplantar a una determinada persona.
- **Análisis demográfico avanzado:** permitiendo con ello obtener la edad, el género y el origen étnico de la persona.
- **Detección de emociones:** permitiendo con ello conocer las posibles emociones de una persona como, por ejemplo, la felicidad, la tristeza, la sorpresa y la ira.

Aunque los avances anteriores son bastante interesantes, existen otros enfoques del reconocimiento facial que están ganando protagonismo en la actualidad.

El robot ***Smart Service***, de la compañía *New Era AI Robotic*, permite tener un interlocutor en tiendas u oficinas de la administración pública que reconoce a las personas y les proporciona información relativa al establecimiento en el que se encuentran [25] [26]:

Ilustración 19: Robot Smart Service

El robot ***Robelf***, de la compañía *Robotelf Technologies*, permite tener un agente de vigilancia en casa capaz de reconocer a los miembros de la familia y a posibles extraños que entren en

la casa (ya que posee sensores de vídeo, voz y posición), además de poder enseñar materias como la programación o el inglés [25] [27]:

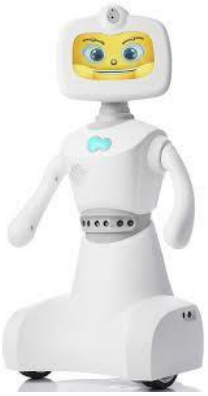


Ilustración 20: Robot Robelf

El robot ***Barista***, de la compañía *Korea Telecom*, permite tener a un camarero que es capaz de reconocer a los usuarios y ofrece a estos obtener un café personalizado en base a las especificaciones que le den al robot [28] [29]:

Ilustración 21: Robot Barista

Por último, el robot ***Kebbi***, de la compañía *Nuwa Robotics*, supone tener a una niñera que permite cuidar niños en casas o guarderías, además de saber cantar, bailar y dar clases de inglés. Aparte de estas funcionalidades, también es capaz de reconocer personas y detectar si entra un extraño en casa cuando se encuentra sólo con los niños [28] [30]:

Ilustración 22: Robot Kebbi

3. OBJETIVOS DEL TRABAJO

El principal objetivo que se pretende conseguir con este Trabajo de Fin de Grado es crear un sistema que, mediante la Planificación Automática y el robot humanoide NAO, permita a dos usuarios intercambiar mensajes entre sí, pudiendo elegir entre dos tipos de modalidad de entrega del mensaje, dependiendo de si el mensaje es personal o no.

La primera modalidad consistirá en que el mensaje que el usuario emisor quiere mandar será enviado por el NAO como un audio adjunto que se mandará al correo electrónico del usuario receptor. Esta será la modalidad escogida en caso de que el mensaje a entregar sea personal.

La segunda modalidad consistirá en que el mensaje que el usuario emisor quiere mandar será entregado por el propio NAO al usuario receptor, realizando con ello un recorrido físico hasta el usuario receptor con dos etapas diferenciadas:

- Realización de una ruta que permita llegar hasta donde se encuentra el usuario receptor.
- Identificación del usuario receptor, mediante reconocimiento facial, y entrega del mensaje.

Esta segunda modalidad, de mayor complejidad, será la escogida en caso de que el mensaje a entregar no sea personal y, a diferencia de la modalidad de entrega por correo, permitirá al usuario receptor del mensaje utilizar al robot para enviar una respuesta al mensaje recibido.

La consecución de un sistema que cumpla estas características y, por tanto, el objetivo principal, se compone de las siguientes fases:

- **Estudio y experimentación con el NAO:** para poder conseguir un sistema como el que se pretende es fundamental conocer cómo funciona el robot NAO, cómo se le pueden mandar acciones para que las realice y qué posibilidades ofrece.
- **Estudio de PELEA:** la arquitectura de PELEA es compleja, por lo que conviene primero entender su funcionamiento para poder posteriormente realizar las modificaciones oportunas.
- **Estudio de PDDL:** debido a que tanto el dominio como el problema con el que se trabaja en PELEA están escritos en PDDL, resulta fundamental entender la notación y complejidad de este lenguaje de Planificación Automática.

- **Definición del dominio y del problema en PDDL:** una vez que se ha comprendido el lenguaje de PDDL, será el momento de modelar el dominio y el problema de la situación que se pretende resolver.
- **Modificación de PELEA:** cuando se han comprendido PELEA y PDDL, habrá que añadir el problema modelado a esta arquitectura y realizar las modificaciones necesarias en ella para su correcto funcionamiento.
- **Creación e implementación del servidor:** dado que las acciones que realiza el NAO están escritas en Python y PELEA está escrito en Java, será necesario separar la ejecución de las acciones de su planificación, creando para ello un servidor programado en Python que recibirá peticiones (acciones) del cliente (PELEA), las realizará y devolverá a dicho cliente las respuestas oportunas (información del estado de la acción).
- **Pruebas del sistema:** comprobación de que la implementación es correcta y el sistema desarrollado realiza bien todas sus funciones.

Una vez que se hayan realizado todas las fases, se pretende tener un sistema que no sólo sea funcional, sino que, en el momento de aumentar la complejidad del problema a resolver, las modificaciones que se deban realizar sean mínimas.

4. ANÁLISIS DEL SISTEMA

El objetivo de este capítulo es realizar el análisis del sistema, que es la obtención de una especificación detallada del sistema a desarrollar. Para ello, se describirán las características funcionales del sistema, se explicará el entorno operacional, se mostrarán los casos de uso, se enumerarán los requisitos que se deben cumplir y se comprobará la correspondencia de los casos de uso y los requisitos mediante una matriz de trazabilidad.

4.1 Descripción de las características funcionales

El análisis completo del sistema a desarrollar se ha podido realizar una vez que la fase de estudio y experimentación con el NAO ha llegado a su fin, ya que con el final de esta fase ya se tiene conocimiento de las capacidades del NAO que pueden ayudar a conseguir los objetivos detallados en el capítulo 3 de este documento.

La primera modalidad de entrega del mensaje, dada su simplicidad, no ha supuesto un problema en el planteamiento de su realización.

Sin embargo, la segunda modalidad sí ha supuesto conocer en profundidad las habilidades del NAO, para así encontrar aquellas que permitieran realizar una ruta física y la identificación de personas.

Para la realización de la ruta física, se descubrió que el NAO tiene un conjunto de marcas que reconoce, denominadas *Naomarks*. Por tanto, el objetivo es crear una ruta compuesta por estas marcas, consiguiendo realizar con ello una navegación guiada por ellas, que servirán como pista de por dónde el NAO debe continuar avanzando.

Para la identificación de personas, se descubrió que el NAO posee una base de datos en la que puede almacenar tanto el nombre como el rostro de una persona, permitiéndole posteriormente reconocer ese rostro y relacionarlo con su nombre asociado.

Una vez que se solventaron las dudas en el planteamiento de la entrega personal del mensaje, se pretende diseñar un sistema que tenga las siguientes funcionalidades:

- **Aprendizaje de rostro:** el sistema permitirá a un usuario almacenar un rostro en la base de datos de rostros del NAO.
- **Búsqueda de rostro:** el sistema permitirá a un usuario buscar un rostro en la base de datos de rostros del NAO.

- **Listado de rostros:** el sistema permitirá a un usuario ver una lista con los rostros almacenados actualmente en la base de datos de rostros del NAO.
- **Eliminación de rostro:** el sistema permitirá a un usuario eliminar un rostro concreto de la base de datos de rostros del NAO.
- **Eliminación de todos los rostros:** el sistema permitirá a un usuario eliminar todos los rostros de la base de datos de rostros del NAO.
- **Entrega de mensaje de usuario emisor a usuario receptor por correo:** el sistema permitirá a un usuario emisor mandar por correo un mensaje como audio adjunto al correo electrónico del usuario receptor.
- **Entrega personal de mensaje de usuario emisor a usuario receptor:** el sistema permitirá a un usuario emisor entregar un mensaje a un usuario receptor haciendo que el NAO sea quién lo entregue personalmente.
- **Entrega de mensaje de usuario receptor a usuario emisor por correo:** el sistema permitirá al usuario receptor, tras haber recibido personalmente del NAO el mensaje del usuario emisor, mandar una respuesta utilizando la entrega por correo.
- **Entrega personal de mensaje de usuario receptor a usuario emisor:** el sistema permitirá al usuario receptor, tras haber recibido personalmente del NAO el mensaje del usuario emisor, mandar una respuesta utilizando de nuevo la entrega personal por parte del NAO.

Dado que se han incluido funciones básicas de una base de datos, se ha determinado proteger el uso del sistema de usuarios no autorizados mediante la utilización de un fichero cifrado en el que se guardarán los usuarios que sí están autorizados para utilizar el sistema, no permitiendo el uso de este a usuarios que no se encuentren en dicho fichero.

Teniendo en cuenta que es peligroso que las funciones relacionadas con los rostros (aprender un rostro, buscar un rostro, listar todos los rostros, eliminar un rostro y eliminar todos los rostros) las pueda realizar cualquier usuario del sistema, estas funciones se restringirán de forma que sólo el usuario administrador las pueda realizar.

4.2 Entorno operacional

Las tecnologías que se han utilizado para realizar la implementación del sistema son las siguientes:

- **Ubuntu:** se trata de un sistema operativo de Linux. Su uso se debe a su menor consumo de recursos, a su mayor velocidad de procesamiento y a la falta de eficiencia de su utilización en una máquina virtual. La versión utilizada en este desarrollo es la versión 18.04, con una arquitectura de 64 bits.
- **PELEA:** se trata de una arquitectura genérica de código libre programada en Java y que se utiliza para la aplicación de la Planificación Automática en los robots. Su uso se debe a la aplicación de la Planificación Automática en este trabajo.
- **Java Development Kit:** se trata de un software que provee herramientas de desarrollo para la creación de programas en Java. Su uso de debe a la utilización de la arquitectura PELEA, que está programada en Java. La versión utilizada en este desarrollo es la versión 10.
- **Eclipse:** se trata de un entorno de desarrollo integrado (IDE) que permite el desarrollo de aplicaciones Java. Su uso de debe a las facilidades que ofrece en la programación del lenguaje Java. La versión utilizada en este desarrollo es la versión Oxygen.
- **Python:** se trata de un lenguaje de programación interpretado. Su uso se debe a que es el lenguaje elegido para la programación de las acciones que realiza el NAO. La versión utilizada en este desarrollo es la versión 2.7.
- **Spyder:** se trata de un entorno de desarrollo interactivo para el lenguaje Python. Su uso se debe a las facilidades que ofrece en la programación del lenguaje Python. La versión utilizada en este desarrollo es la versión 3.2.6.
- **Choregraphe:** se trata de una aplicación de escritorio multiplataforma que permite la monitorización y el control del robot humanoide NAO. Su uso se debe a la utilización del NAO en este trabajo. La versión utilizada en este desarrollo es la versión 2.1.4.
- **GNU Privacy Guard:** se trata de una herramienta de cifrado y firmas digitales. Su uso se debe al cifrado del fichero de usuarios autorizados para utilizar el sistema, con el objetivo de aumentar la seguridad del sistema. La versión utilizada en este desarrollo es la versión 2.2.4.

4.3 Especificación de casos de uso

Un caso de uso consiste en una interacción entre el usuario y el sistema, donde se especifican los pasos necesarios a seguir para poder realizar una tarea específica.

Para poder describir los casos de uso es fundamental realizar dos tareas. En primer lugar, se deben identificar los actores que van a intervenir en las interacciones que se produzcan en los casos de uso.

Una vez que los actores que van a interactuar han sido identificados, se debe decidir cómo se van a querer representar los casos de uso para poder describirlos de la mejor manera posible.

4.3.1 Identificación de actores

En la representación de los casos de uso es muy importante identificar los actores u entidades que con su interactuación van a permitir la realización de una tarea específica:

- **Usuario emisor:** representa a la persona que envía un mensaje al usuario receptor.
- **Usuario receptor:** representa a la persona que recibe el mensaje del usuario emisor.
- **Administrador:** representa a la persona que instalará el sistema, configurará el fichero de usuarios autorizados y el fichero PDDL del problema y utilizará el sistema.

4.3.2 Representación de casos de uso

Los casos de uso van a ser representados de forma tabular, donde se utilizarán diferentes campos para realizar una completa descripción de cada caso de uso.

Para describir cada uno de los casos de uso, se utilizará el siguiente modelo de tabla:

CU-XXX		
Nombre		
Descripción		
Fecha		
Secuencia normal	**Paso**	**Acción**
	1	
	2	
	3	
Precondiciones		
Postcondiciones		

Tabla 1: Formato de caso de uso

En la tabla anterior, podemos observar los siguientes campos:

- **Identificador:** código único que identifica al caso de uso y que se compone de las siguientes partes:
  - **CU:** indica que se trata de un caso de uso.
  - **XXX:** indica el número único (identificador) que tiene el caso de uso. El número de cada caso de uso no tiene relación con su importancia.
- **Nombre:** título que da una breve información acerca del objetivo del caso de uso.
- **Descripción:** explicación detallada del caso de uso.
- **Fecha:** fecha de la última modificación del caso de uso.
- **Secuencia normal:** explicación de los pasos que se realizan en el caso de uso.
- **Precondiciones:** condiciones que deben cumplirse al principio de la secuencia para que se pueda realizar el caso de uso.
- **Postcondiciones:** consecuencias que tiene la finalización del caso de uso.

4.3.2.1 Casos de uso

Tal y como se ha especificado, el usuario emisor del mensaje no puede realizar funciones relacionadas con la gestión de rostros, por lo que tiene los siguientes casos de uso:

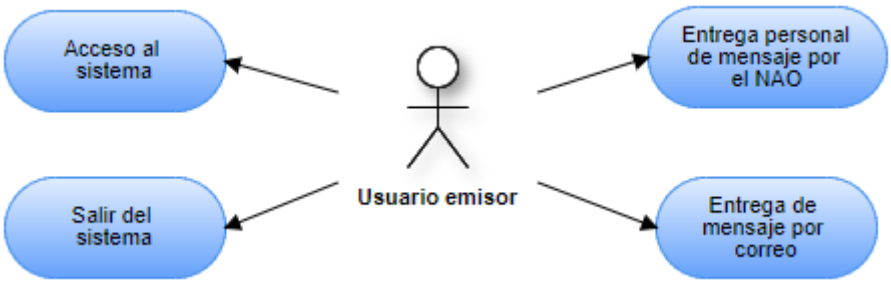


Ilustración 23: Casos de uso del usuario emisor

En el caso del usuario receptor, sucede lo mismo que con el usuario emisor, sólo que en este caso este usuario no inicia el sistema ya que cuando se comunica con el NAO el sistema ya ha sido iniciado:

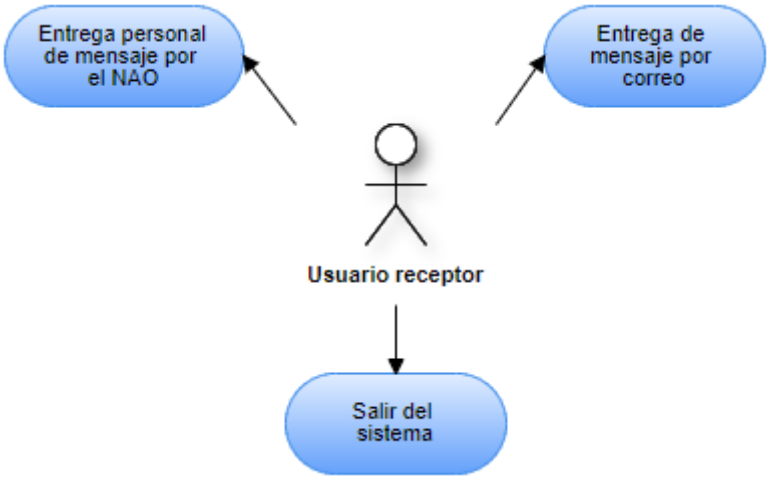


Ilustración 24: Casos de uso del usuario receptor

En el caso del administrador, puede realizar todas las funciones del sistema, por lo que puede realizar los siguientes casos de uso:

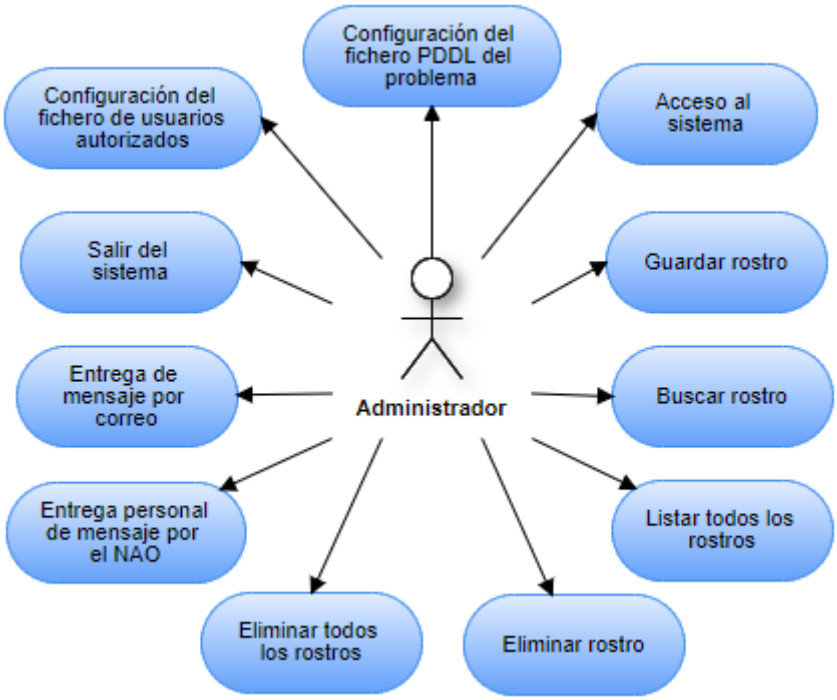


Ilustración 25: Casos de uso del administrador

A continuación, se muestran de forma tabular los casos de uso identificados:

CU-001		
Nombre	Configuración de usuarios autorizados	
Descripción	El administrador configura el fichero de usuarios autorizados para utilizar el sistema.	
Fecha	03/03/2019	
Secuencia normal	**Paso**	**Acción**
	1	El administrador abre el fichero de usuarios autorizados del sistema.
	2	El administrador añade el nombre, el correo y el rol de cada uno de los usuarios que estarán autorizados para utilizar el sistema.
	3	El administrador guarda los cambios del fichero de usuarios autorizados del sistema.
	4	El administrador cierra el fichero de usuarios autorizados del sistema.
	5	El administrador cifra el fichero de usuarios autorizados del sistema.
Precondiciones	El administrador debe haber descargado la carpeta con el código del Trabajo de Fin de Grado.	
Postcondiciones	El fichero de usuarios autorizados para utilizar el sistema se encontrará configurado y cifrado.	

Tabla 2: CU-001, Configuración de usuarios autorizados

CU-002		
Nombre	Configuración del problema en PDDL	
Descripción	El administrador configura en el fichero PDDL del problema el mapa de marcas que el NAO utilizará para entregar un mensaje personalmente.	
Fecha	03/03/2019	
Secuencia normal	**Paso**	**Acción**
	1	El administrador abre el fichero PDDL del problema.
	2	El administrador configura el mapa de marcas que el NAO utilizará.
	3	El administrador guarda los cambios del fichero del problema.
	4	El administrador cierra el fichero del problema.
Precondiciones	El administrador debe haber configurado el fichero de usuarios autorizados.	
Postcondiciones	El sistema puede entregar un mensaje utilizando al NAO para recorrer una ruta de marcas que reconoce.	

Tabla 3: CU-002, Configuración del problema en PDDL

CU-003		
Nombre	Acceso al sistema	
Descripción	Un usuario accede al sistema para poder realizar alguna de sus funciones.	
Fecha	03/03/2019	
Secuencia normal	**Paso**	**Acción**
	1	El usuario inicia el sistema.
	2	El sistema le pide al usuario que coloque su rostro delante de la cara del NAO.
	3	El usuario coloca su rostro delante de la cara del NAO.
	4	El sistema comprueba que es un usuario autorizado y con el rostro guardado en la base de datos del NAO.
	5	El sistema informa al usuario de su rol y de que está autorizado.
	6	El sistema pregunta al usuario que función quiere realizar.
Precondiciones	El administrador debe haber configurado el fichero de usuarios autorizados.	
	El usuario debe estar a menos de cincuenta centímetros del NAO.	
	El usuario debe estar guardado en el fichero de usuarios autorizados.	
	El usuario debe tener guardado su rostro en la base de datos del NAO.	
	El NAO debe de estar encendido.	
Postcondiciones	El usuario podrá realizar una función de las que permite el sistema.	

Tabla 4: CU-003, Acceso al sistema

CU-004		
Nombre	Aprendizaje de rostro	
Descripción	El administrador quiere guardar un rostro en la base de datos del NAO.	
Fecha	03/03/2019	
Secuencia normal	**Paso**	**Acción**
	1-6	Los mismos que en el CU-003.
	7	El administrador le dice al sistema la función de guardar rostro.
	8	El sistema le pide al administrador que le diga el nombre y los apellidos del rostro que quiere guardar.
	9	El administrador le dice al sistema el nombre y los apellidos del rostro que quiere guardar.
	10	El sistema comprueba que el usuario dicho es un usuario autorizado y que no tiene el rostro guardado en la base de datos del NAO.
	11	El sistema le pide al administrador que el usuario del que se quiere guardar el rostro ponga delante de la cara del NAO su rostro.
	12	El usuario en cuestión pone delante de la cara del NAO su rostro.
	13	El sistema almacena el nombre y el rostro en la base de datos del NAO.

	14	El sistema informa al administrador de que el rostro ha sido almacenado.
Precondiciones	El rostro del usuario no debe estar ya guardado en la base de datos del NAO.	
Postcondiciones	El rostro del usuario del que se quiere guardar el rostro habrá sido almacenado en la base de datos del NAO.	
	El sistema terminará su ejecución.	

Tabla 5: CU-004, Aprendizaje de rostro

CU-005		
Nombre	Búsqueda de rostro	
Descripción	El administrador quiere buscar si un usuario tiene guardado su rostro en la base de datos del NAO.	
Fecha	03/03/2019	
Secuencia normal	**Paso**	**Acción**
	1-6	Los mismos que en el CU-003.
	7	El administrador le dice al sistema la función de buscar rostro.
	8	El sistema comprueba que hay al menos un rostro almacenado en la base de datos del NAO.
	9	El sistema le pide al administrador que le diga el nombre y los apellidos del rostro que quiere buscar.
	10	El administrador le dice al sistema el nombre y los apellidos del rostro que quiere buscar.
	11	El sistema realiza la búsqueda del nombre en la base de datos del NAO.
	12	El sistema informa al usuario emisor si el rostro del nombre que ha dicho está en la base de datos del NAO o no.
Precondiciones	La base de datos del NAO debe de tener al menos un rostro almacenado.	
Postcondiciones	El administrador sabrá si el rostro del que ha dicho el nombre está almacenado en la base de datos del NAO o no lo está.	
	El sistema terminará su ejecución.	

Tabla 6: CU-005, Búsqueda de rostro

CU-006		
Nombre	Listado de rostros	
Descripción	El administrador quiere ver todos los usuarios cuyos rostros están almacenados en la base de datos del NAO.	
Fecha	03/03/2019	
Secuencia normal	**Paso**	**Acción**
	1-6	Los mismos que en el CU-003.
	7	El administrador le dice al sistema la función de listar rostros.
	8	El sistema comprueba que hay al menos un rostro almacenado en la base de datos del NAO.

	9	El sistema informa al administrador de que le va a mostrar la lista de usuarios cuyos rostros están almacenados en la base de datos del NAO.
	10	El sistema imprime en su consola la lista de usuarios.
	11	El sistema informa al administrador de que la lista de rostros está disponible en la consola.
Precondiciones	La base de datos del NAO debe de tener al menos un rostro almacenado.	
Postcondiciones	El administrador podrá ver la lista de usuarios con rostros almacenados en la base de datos del NAO en la consola del sistema.	
	El sistema terminará su ejecución.	

Tabla 7: CU-006, Listado de rostros

CU-007		
Nombre	Eliminación de rostro	
Descripción	El administrador quiere eliminar un rostro de la base de datos del NAO.	
Fecha	03/03/2019	
Secuencia normal	**Paso**	**Acción**
	1-6	Los mismos que en el CU-003.
	7	El administrador le dice al sistema la función de eliminar rostro.
	8	El sistema comprueba que hay al menos un rostro almacenado en la base de datos del NAO.
	9	El sistema le pide al administrador que le diga el nombre y los apellidos del rostro que quiere eliminar.
	10	El administrador le dice al sistema el nombre y los apellidos del rostro que quiere eliminar.
	11	El sistema realiza la búsqueda del nombre en la base de datos del NAO.
	12	El sistema elimina el nombre y el rostro de la base de datos del NAO.
	13	El sistema informa al administrador de que el rostro ha sido eliminado.
Precondiciones	La base de datos del NAO debe de tener al menos un rostro almacenado.	
	El nombre y el rostro comunicados por el administrador deben estar almacenados en la base de datos del NAO.	
Postcondiciones	El rostro que el administrador quería eliminar habrá sido eliminado de la base de datos del NAO.	
	El sistema terminará su ejecución.	

Tabla 8: CU-007, Eliminación de rostro

CU-008		
Nombre	Eliminación de todos los rostros	
Descripción	El administrador quiere eliminar todos los rostros de la base de datos del NAO.	
Fecha	03/03/2019	
Secuencia normal	**Paso**	**Acción**
	1-6	Los mismos que en el CU-003.
	7	El administrador le dice al sistema la función de eliminar todos los rostros.
	8	El sistema comprueba que hay al menos un rostro almacenado en la base de datos del NAO.
	9	El sistema informa al administrador de que va a eliminar todos los rostros almacenados de la base de datos del NAO.
	10	El sistema elimina todos los rostros de la base de datos del NAO.
	11	El sistema informa al administrador de que todos los rostros han sido eliminados de la base de datos del NAO.
Precondiciones	La base de datos del NAO debe de tener al menos un rostro almacenado.	
Postcondiciones	Todos los rostros que el sistema tenía almacenados en la base de datos del NAO habrán sido eliminados.	
	El sistema terminará su ejecución.	

Tabla 9: CU-008, Eliminación de todos los rostros

CU-009		
Nombre	Entrega de un mensaje del usuario emisor por correo electrónico	
Descripción	El usuario emisor quiere mandar un mensaje al usuario receptor utilizando el correo electrónico.	
Fecha	03/03/2019	
Secuencia normal	**Paso**	**Acción**
	1-6	Los mismos que en el CU-003.
	7	El usuario emisor le dice al sistema la función de comunicarse.
	8	El sistema le pide al usuario emisor que le diga el nombre y los apellidos del usuario receptor.
	9	El usuario emisor le dice al sistema el nombre y los apellidos del usuario receptor.
	10	El sistema comprueba que el usuario receptor es un usuario autorizado.
	11	El sistema comprueba si el usuario receptor tiene su rostro guardado en la base de datos del NAO.
	12	El sistema informa al usuario emisor de las opciones de duración que puede tener el mensaje que va a mandar.
	13	El sistema pregunta al usuario emisor qué opción de duración quiere utilizar.
	14	El usuario emisor le dice al sistema una de las opciones de

		duración.
	15	El sistema informa al usuario emisor de la opción que ha elegido.
	16	El sistema le pide al usuario emisor que empiece a decir su mensaje.
	17	El sistema empieza a grabar el audio.
	18	El usuario emisor dice su mensaje.
	19	El sistema deja de grabar el audio.
	20	El sistema informa al usuario emisor de que ya ha grabado su mensaje.
	21	El sistema pregunta al usuario emisor si el mensaje es personal.
	22	El usuario emisor le dice al sistema que el mensaje es personal.
	23	El sistema informa al usuario emisor de que el mensaje es personal y va a transmitirlo por correo.
	24	El sistema manda el correo electrónico.
	25	El sistema informa al usuario emisor de que el mensaje se ha mandado correctamente si el correo electrónico del usuario receptor es un correo existente.
Precondiciones	El usuario receptor debe estar guardado en el fichero de usuarios autorizados.	
	El correo electrónico del usuario receptor debe ser un correo existente.	
Postcondiciones	El usuario receptor habrá recibido en su correo electrónico el mensaje que le ha dejado el usuario emisor.	
	El sistema terminará su ejecución.	

Tabla 10: CU-009, Entrega de un mensaje del usuario emisor por correo electrónico

CU-010		
Nombre	Entrega física de un mensaje del usuario emisor	
Descripción	El usuario emisor quiere mandar un mensaje al usuario receptor haciendo que sea el NAO quien lleve el mensaje personalmente.	
Fecha	03/03/2019	
Secuencia normal	**Paso**	**Acción**
	1-20	Los mismos que en el CU-009.
	21	El sistema pregunta al usuario emisor si el mensaje es personal.
	22	El usuario emisor le dice al sistema que el mensaje no es personal.
	23	El sistema informa al usuario emisor de que el mensaje no es personal y lo va a transmitir el propio NAO.
	24	El sistema utiliza al NAO para realizar un recorrido físico guiado por marcas que el NAO reconoce, hasta llegar a alcanzar la última marca.
	25	El sistema informa de que ha llegado a la última marca de su

		ruta.
	26	El sistema busca al usuario receptor.
	27	El sistema encuentra a una persona.
	28	El sistema pide a esa persona que mire a la cara del NAO.
	29	Dicha persona mira al NAO a la cara.
	30	En caso de que la persona sea el usuario receptor, el sistema saluda al usuario receptor y le informa de quién es el mensaje que le va a entregar.
	31	El sistema reproduce el mensaje grabado.
	32	El sistema pregunta al usuario receptor si quiere responder al mensaje.
	33	El usuario receptor le dice al NAO que no quiere responder al mensaje.
Precondiciones	El administrador debe haber configurado el mapa de marcas que el NAO utiliza para su recorrido.	
	El usuario receptor debe tener guardado su rostro en la base de datos del NAO.	
	El usuario receptor debe estar en una marca de la ruta que sigue el NAO.	
	La precondición del CU-009 de que el correo electrónico del usuario receptor debe ser un correo existente no es necesaria en este caso de uso.	
Postcondiciones	El usuario receptor habrá recibido del propio NAO el mensaje que le ha dejado el usuario emisor.	
	El sistema terminará su ejecución.	

Tabla 11: CU-010, Entrega física de un mensaje del usuario emisor

CU-011		
Nombre	Entrega de un mensaje del usuario receptor por correo electrónico	
Descripción	El usuario receptor quiere responder al usuario emisor el mensaje recibido con otro mensaje mandándolo por correo.	
Fecha	03/03/2019	
Secuencia normal	**Paso**	**Acción**
	1-31	Los mismos que en el CU-010.
	32	El sistema pregunta al usuario receptor si quiere responder al mensaje.
	33	El usuario receptor le dice al sistema que quiere responder al mensaje.
	34-47	Se repiten los pasos del 12 al 25 (incluidos) del CU-009, sólo que en este caso de uso el usuario que interactúa con el NAO es el usuario receptor y no el usuario emisor del mensaje, mandándose el mensaje al correo electrónico del usuario emisor.
Precondiciones	El correo electrónico del usuario emisor debe ser un correo existente.	
	La precondición del CU-009 de que el correo electrónico del usuario receptor debe ser un correo existente no es necesaria en este caso de uso.	

Postcondiciones	El usuario receptor habrá recibido el mensaje del usuario emisor y el usuario emisor habrá recibido la respuesta en su correo electrónico.
	El sistema terminará su ejecución.

Tabla 12: CU-011, Entrega de un mensaje del usuario receptor por correo electrónico

CU-012		
Nombre	Entrega física de un mensaje del usuario receptor	
Descripción	El usuario receptor quiere responder al usuario emisor el mensaje recibido con otro mensaje haciendo que sea el NAO quien lleve el mensaje personalmente.	
Fecha	03/03/2019	
Secuencia normal	**Paso**	**Acción**
	1-42	Los mismos que en el CU-011 (el paso 42 del CU-011 es el mismo paso que el paso 20 del CU-009, con los cambios explicados en el CU-011).
	43-55	Se repiten los pasos del 21 al 33 (incluidos) del CU-010, sólo que en los casos donde actúa el usuario emisor ahora es el usuario receptor y viceversa.
Precondiciones	El usuario emisor debe estar en la primera marca de la ruta que siguió el NAO cuando entregó el mensaje al usuario receptor, ya que ahora el NAO realizará dicha ruta en sentido contrario.	
	La precondición del CU-009 de que el correo electrónico del usuario receptor debe ser un correo existente no es necesaria en este caso de uso.	
	La precondición del CU-011 de que el correo electrónico del usuario emisor debe ser un correo existente no es necesaria en este caso de uso.	
Postcondiciones	El usuario receptor habrá recibido el mensaje del usuario emisor y el usuario emisor habrá recibido del propio NAO la respuesta al mensaje.	
	El sistema terminará su ejecución.	

Tabla 13: CU-012, Entrega física de un mensaje del usuario receptor

CU-013		
Nombre	Salida del sistema	
Descripción	Un usuario decide elegir como función a realizar la de salir del sistema.	
Fecha	03/03/2019	
Secuencia normal	**Paso**	**Acción**
	1-6	Los mismos que en el CU-003.
	7	El usuario le dice al sistema la función de salir.
	8	El sistema termina su ejecución.
Precondiciones	Sus únicas precondiciones son las del CU-003.	
Postcondiciones	El sistema terminará su ejecución.	

Tabla 14: CU-013, Salida del sistema

4.4 Especificación de requisitos

A continuación, se especifican cada uno de los requisitos que el sistema debe cumplir, tanto los que se basan en las funciones que va a realizar el sistema como los que se basan en la forma en que se van a llevar a cabo dichas funciones.

Para ello, se distingue entre dos tipos de requisitos:

- **Requisitos funcionales o de capacidad:** indican los servicios que debe realizar el sistema y el comportamiento que tendrá este ante entradas particulares.
- **Requisitos no funcionales o de restricción**: indican las restricciones que debe cumplir el sistema o el proceso que se emplee para desarrollarlo.

Para describir cada uno de los requisitos, se utilizará el siguiente modelo de tabla:

RXX-YYY			
Nombre			
Descripción			
Fecha		**Estabilidad**	☐ Alta ☐ Media ☐ Baja
Prioridad	☐ Alta ☐ Media ☐ Baja	**Claridad**	☐ Alta ☐ Media ☐ Baja
Necesidad	☐ Esencial ☐ Deseable ☐ Opcional	**Verificabilidad**	☐ Alta ☐ Media ☐ Baja

Tabla 15: Formato de requisito

En la tabla anterior, podemos observar los siguientes campos:

- **Identificador:** código único que identifica al requisito y que se compone de las siguientes partes:
  - **R:** indica que se trata de un requisito.
  - **XX:** indica a cuál de los dos grupos de requisitos pertenece:
    - **F:** funcional.
    - **NF:** no funcional.
  - **YYY:** indica el número único (identificativo) que tiene el requisito. El número de cada requisito no tiene relación con su importancia.
- **Nombre:** título que da una breve información acerca del objetivo del requisito.

- **Descripción:** explicación detallada del requisito.
- **Fecha:** fecha de la última modificación del requisito.
- **Estabilidad:** indica la probabilidad de que el requisito sufra posibles modificaciones:
  - **Alta:** el requisito es estable y es muy poco probable que cambie.
  - **Media:** el requisito no es estable pero no tiene por qué sufrir modificaciones.
  - **Baja:** el requisito no es estable y sufrirá modificaciones.
- **Prioridad:** indica el orden temporal del requisito:
  - **Alta:** el requisito debe abordarse desde el principio del desarrollo del sistema.
  - **Media:** el requisito debe abordarse cuando se hayan cumplido todos los de prioridad alta.
  - **Baja:** el requisito debe abordarse sólo cuando se hayan cumplido todos los de prioridad alta y media.
- **Claridad:** indica la ambigüedad del requisito:
  - **Alta:** el requisito es claro al expresarse con un vocabulario preciso.
  - **Media:** el requisito tiene aspectos en su descripción que en ciertos momentos pueden llevar a confusión.
  - **Baja:** el requisito es ambiguo y con seguridad provocará confusiones.
- **Necesidad:** indica la importancia del requisito:
  - **Esencial:** resulta fundamental que el sistema cumpla el requisito.
  - **Deseable:** es importante cumplirlo una vez que todos los de alta prioridad hayan sido cubiertos.
  - **Opcional:** no se considera especialmente relevante la ausencia de este requisito.
- **Verificabilidad:** indica el grado de facilidad con la que se puede comprobar que el sistema cumple este requisito:
  - **Alta:** resulta sencilla la comprobación de que el requisito se cumple.
  - **Media:** se puede comprobar que el requisito se cumple, pero puede haber problemas para determinarlo.
  - **Baja:** se presentan muchas dificultades para la verificación del requisito.

4.4.1 Requisitos funcionales

Como ya se ha explicado, los requisitos funcionales definen lo que el sistema debe realizar. A continuación, se detalla la lista de requisitos funcionales identificados:

RF-001			
Nombre	Comprobación de usuario autorizado por fichero		
Descripción	El sistema podrá comprobar que un usuario determinado está guardado en el fichero de los usuarios autorizados para utilizar el sistema, pudiendo obtener su nombre, su correo y su rol.		
Fecha	03/03/2019	**Estabilidad**	☒ Alta ☐ Media ☐ Baja
Prioridad	☒ Alta ☐ Media ☐ Baja	**Claridad**	☒ Alta ☐ Media ☐ Baja
Necesidad	☒ Esencial ☐ Deseable ☐ Opcional	**Verificabilidad**	☒ Alta ☐ Media ☐ Baja

Tabla 16: RF-001, Comprobación de usuario autorizado por fichero

RF-002			
Nombre	Detección de rostro		
Descripción	El sistema podrá detectar cuando hay un rostro delante de la cara del NAO.		
Fecha	03/03/2019	**Estabilidad**	☒ Alta ☐ Media ☐ Baja
Prioridad	☒ Alta ☐ Media ☐ Baja	**Claridad**	☒ Alta ☐ Media ☐ Baja
Necesidad	☒ Esencial ☐ Deseable ☐ Opcional	**Verificabilidad**	☒ Alta ☐ Media ☐ Baja

Tabla 17: RF-002, Detección de rostro

RF-003	
Nombre	Reconocimiento de rostro
Descripción	El sistema podrá reconocer un rostro que esté almacenado en la base de

	datos del NAO.		
Fecha	03/03/2019	**Estabilidad**	☒ Alta ☐ Media ☐ Baja
Prioridad	☒ Alta ☐ Media ☐ Baja	**Claridad**	☒ Alta ☐ Media ☐ Baja
Necesidad	☒ Esencial ☐ Deseable ☐ Opcional	**Verificabilidad**	☒ Alta ☐ Media ☐ Baja

Tabla 18: RF-003, Reconocimiento de rostro

RF-004			
Nombre	Formato de correos del fichero de usuarios autorizados		
Descripción	El sistema podrá comprobar que el correo de cada uno de los usuarios del fichero de los usuarios autorizados para utilizar el sistema tiene un formato correcto.		
Fecha	03/03/2019	**Estabilidad**	☒ Alta ☐ Media ☐ Baja
Prioridad	☒ Alta ☐ Media ☐ Baja	**Claridad**	☒ Alta ☐ Media ☐ Baja
Necesidad	☒ Esencial ☐ Deseable ☐ Opcional	**Verificabilidad**	☒ Alta ☐ Media ☐ Baja

Tabla 19: RF-004, Formato de correos del fichero de usuarios autorizados

RF-005			
Nombre	Usuarios existentes en el fichero de usuarios autorizados		
Descripción	El sistema podrá comprobar que hay al menos un usuario guardado en el fichero de usuarios autorizados.		
Fecha	03/03/2019	**Estabilidad**	☒ Alta ☐ Media ☐ Baja
Prioridad	☒ Alta ☐ Media ☐ Baja	**Claridad**	☒ Alta ☐ Media ☐ Baja

Necesidad	☒ Esencial ☐ Deseable ☐ Opcional	**Verificabilidad**	☒ Alta ☐ Media ☐ Baja

Tabla 20: RF-005, Usuarios existentes en el fichero de usuarios autorizados

RF-006			
Nombre	Resolución de problemas en PDDL		
Descripción	El sistema podrá crear un plan de acciones de alto nivel a realizar a partir de dos ficheros PDDL, uno con el dominio y otro con el problema.		
Fecha	03/03/2019	**Estabilidad**	☒ Alta ☐ Media ☐ Baja
Prioridad	☒ Alta ☐ Media ☐ Baja	**Claridad**	☒ Alta ☐ Media ☐ Baja
Necesidad	☒ Esencial ☐ Deseable ☐ Opcional	**Verificabilidad**	☒ Alta ☐ Media ☐ Baja

Tabla 21: RF-006, Resolución de problemas en PDDL

RF-007			
Nombre	Conversión de acciones		
Descripción	El sistema podrá traducir las acciones de alto nivel del plan creado a acciones de bajo nivel que se ejecutarán en el NAO.		
Fecha	03/03/2019	**Estabilidad**	☒ Alta ☐ Media ☐ Baja
Prioridad	☒ Alta ☐ Media ☐ Baja	**Claridad**	☒ Alta ☐ Media ☐ Baja
Necesidad	☒ Esencial ☐ Deseable ☐ Opcional	**Verificabilidad**	☒ Alta ☐ Media ☐ Baja

Tabla 22: RF-007, Conversión de acciones

RF-008			
Nombre	Obtención del estado de la acción		
Descripción	El sistema podrá obtener información sobre el estado de una acción de bajo nivel tras la realización de dicha acción por el NAO.		
Fecha	03/03/2019	**Estabilidad**	☒ Alta ☐ Media ☐ Baja
Prioridad	☒ Alta ☐ Media ☐ Baja	**Claridad**	☒ Alta ☐ Media ☐ Baja
Necesidad	☒ Esencial ☐ Deseable ☐ Opcional	**Verificabilidad**	☒ Alta ☐ Media ☐ Baja

Tabla 23: RF-008, Obtención del estado de la acción

RF-009			
Nombre	Replanificación		
Descripción	El sistema podrá crear un nuevo plan de acciones de alto nivel en función de la información recibida sobre el estado de una acción de bajo nivel.		
Fecha	03/03/2019	**Estabilidad**	☒ Alta ☐ Media ☐ Baja
Prioridad	☒ Alta ☐ Media ☐ Baja	**Claridad**	☒ Alta ☐ Media ☐ Baja
Necesidad	☒ Esencial ☐ Deseable ☐ Opcional	**Verificabilidad**	☒ Alta ☐ Media ☐ Baja

Tabla 24: RF-009, Replanificación

RF-010			
Nombre	Capacidad de hablar		
Descripción	El sistema podrá reproducir palabras, que le permitirá decir frases completas.		
Fecha	03/03/2019	**Estabilidad**	☒ Alta ☐ Media ☐ Baja
	☒ Alta		☒ Alta

Prioridad	☐ Media ☐ Baja	**Claridad**	☐ Media ☐ Baja
Necesidad	☒ Esencial ☐ Deseable ☐ Opcional	**Verificabilidad**	☒ Alta ☐ Media ☐ Baja

Tabla 25: RF-010, Capacidad de hablar

RF-011			
Nombre	Reconocimiento de palabras		
Descripción	El sistema podrá reconocer del habla de una persona un grupo de palabras especificado previamente.		
Fecha	03/03/2019	**Estabilidad**	☒ Alta ☐ Media ☐ Baja
Prioridad	☒ Alta ☐ Media ☐ Baja	**Claridad**	☒ Alta ☐ Media ☐ Baja
Necesidad	☒ Esencial ☐ Deseable ☐ Opcional	**Verificabilidad**	☒ Alta ☐ Media ☐ Baja

Tabla 26: RF-011, Reconocimiento de palabras

RF-012			
Nombre	Aprendizaje de rostro		
Descripción	El sistema podrá guardar el nombre y el rostro de una persona en la base de datos del NAO, siempre que no estén ya guardados en ella.		
Fecha	03/03/2019	**Estabilidad**	☒ Alta ☐ Media ☐ Baja
Prioridad	☒ Alta ☐ Media ☐ Baja	**Claridad**	☒ Alta ☐ Media ☐ Baja
Necesidad	☒ Esencial ☐ Deseable ☐ Opcional	**Verificabilidad**	☒ Alta ☐ Media ☐ Baja

Tabla 27: RF-012, Aprendizaje de rostro

RF-013			
Nombre	Comprobación de rostros existentes		
Descripción	El sistema podrá comprobar si hay al menos un rostro guardado en la base de datos del NAO.		
Fecha	03/03/2019	**Estabilidad**	☒ Alta ☐ Media ☐ Baja
Prioridad	☒ Alta ☐ Media ☐ Baja	**Claridad**	☒ Alta ☐ Media ☐ Baja
Necesidad	☒ Esencial ☐ Deseable ☐ Opcional	**Verificabilidad**	☒ Alta ☐ Media ☐ Baja

Tabla 28: RF-013, Comprobación de rostros existentes

RF-014			
Nombre	Búsqueda de rostro		
Descripción	El sistema podrá buscar si un nombre (rostro) está guardado en la base de datos del NAO o no, siempre que esta tenga al menos un nombre (rostro) almacenado.		
Fecha	03/03/2019	**Estabilidad**	☒ Alta ☐ Media ☐ Baja
Prioridad	☒ Alta ☐ Media ☐ Baja	**Claridad**	☒ Alta ☐ Media ☐ Baja
Necesidad	☒ Esencial ☐ Deseable ☐ Opcional	**Verificabilidad**	☒ Alta ☐ Media ☐ Baja

Tabla 29: RF-014, Búsqueda de rostro

RF-015			
Nombre	Listado de rostros		
Descripción	El sistema podrá mostrar en una consola la lista de nombres (rostros) que hay almacenados en la base de datos del NAO, siempre que esta tenga al menos un nombre (rostro) almacenado.		
Fecha	03/03/2019	**Estabilidad**	☒ Alta ☐ Media

			☐ Baja
Prioridad	☒ Alta ☐ Media ☐ Baja	**Claridad**	☒ Alta ☐ Media ☐ Baja
Necesidad	☒ Esencial ☐ Deseable ☐ Opcional	**Verificabilidad**	☒ Alta ☐ Media ☐ Baja

Tabla 30: RF-015, Listado de rostros

RF-016			
Nombre	Eliminación de rostro		
Descripción	El sistema podrá eliminar el nombre y el rostro de una persona de la base de datos del NAO, siempre que esta tenga al menos un nombre (rostro) almacenado y tenga guardado el nombre y el rostro a eliminar.		
Fecha	03/03/2019	**Estabilidad**	☒ Alta ☐ Media ☐ Baja
Prioridad	☒ Alta ☐ Media ☐ Baja	**Claridad**	☒ Alta ☐ Media ☐ Baja
Necesidad	☒ Esencial ☐ Deseable ☐ Opcional	**Verificabilidad**	☒ Alta ☐ Media ☐ Baja

Tabla 31: RF-016, Eliminación de rostro

RF-017			
Nombre	Eliminación de todos los rostros		
Descripción	El sistema podrá eliminar todos los nombres y rostros almacenados en la base de datos del NAO, siempre que esta tenga al menos un nombre (rostro) almacenado.		
Fecha	03/03/2019	**Estabilidad**	☒ Alta ☐ Media ☐ Baja
Prioridad	☒ Alta ☐ Media ☐ Baja	**Claridad**	☒ Alta ☐ Media ☐ Baja
Necesidad	☒ Esencial ☐ Deseable	**Verificabilidad**	☒ Alta ☐ Media

	☐ Opcional		☐ Baja

Tabla 32: RF-017, Eliminación de todos los rostros

RF-018			
Nombre	Grabación de mensaje		
Descripción	El sistema podrá grabar el mensaje de una persona.		
Fecha	03/03/2019	**Estabilidad**	☒ Alta ☐ Media ☐ Baja
Prioridad	☒ Alta ☐ Media ☐ Baja	**Claridad**	☒ Alta ☐ Media ☐ Baja
Necesidad	☒ Esencial ☐ Deseable ☐ Opcional	**Verificabilidad**	☒ Alta ☐ Media ☐ Baja

Tabla 33: RF-018, Grabación de mensaje

RF-019			
Nombre	Determinación del tipo de mensaje		
Descripción	El sistema podrá determinar si un mensaje a entregar es personal o no en función de la respuesta del usuario emisor.		
Fecha	03/03/2019	**Estabilidad**	☒ Alta ☐ Media ☐ Baja
Prioridad	☒ Alta ☐ Media ☐ Baja	**Claridad**	☒ Alta ☐ Media ☐ Baja
Necesidad	☒ Esencial ☐ Deseable ☐ Opcional	**Verificabilidad**	☒ Alta ☐ Media ☐ Baja

Tabla 34: RF-019, Determinación del tipo de mensaje

RF-020			
Nombre	Entrega de mensaje		
Descripción	El sistema podrá entregar un mensaje por correo si este es personal y utilizando al NAO si el mensaje no es personal.		
			☒ Alta

Fecha	03/03/2019	**Estabilidad**	☐ Media ☐ Baja
Prioridad	☒ Alta ☐ Media ☐ Baja	**Claridad**	☒ Alta ☐ Media ☐ Baja
Necesidad	☒ Esencial ☐ Deseable ☐ Opcional	**Verificabilidad**	☒ Alta ☐ Media ☐ Baja

Tabla 35: RF-020, Entrega de mensaje

RF-021			
Nombre	Cuenta de correo propia		
Descripción	El sistema podrá utilizar una cuenta de correo propia cuya función será el envío de correos.		
Fecha	03/03/2019	**Estabilidad**	☒ Alta ☐ Media ☐ Baja
Prioridad	☒ Alta ☐ Media ☐ Baja	**Claridad**	☒ Alta ☐ Media ☐ Baja
Necesidad	☒ Esencial ☐ Deseable ☐ Opcional	**Verificabilidad**	☒ Alta ☐ Media ☐ Baja

Tabla 36: RF-021, Cuenta de correo propia

RF-022			
Nombre	Envío de correo		
Descripción	El sistema podrá mandar desde su cuenta propia de correo un correo electrónico con el audio del mensaje del usuario emisor adjuntado en él al correo del usuario receptor.		
Fecha	03/03/2019	**Estabilidad**	☒ Alta ☐ Media ☐ Baja
Prioridad	☒ Alta ☐ Media ☐ Baja	**Claridad**	☒ Alta ☐ Media ☐ Baja
Necesidad	☒ Esencial	**Verificabilidad**	☒ Alta

	☐ Deseable ☐ Opcional		☐ Media ☐ Baja

Tabla 37: RF-022, Envío de correo

RF-023			
Nombre	Entrega de mensaje por el NAO		
Descripción	El sistema podrá mandar al NAO a entregar un mensaje si el usuario receptor de dicho mensaje tiene guardado su rostro en la base de datos del NAO, no siendo necesario en caso de que el mensaje se mande por correo.		
Fecha	03/03/2019	**Estabilidad**	☒ Alta ☐ Media ☐ Baja
Prioridad	☒ Alta ☐ Media ☐ Baja	**Claridad**	☒ Alta ☐ Media ☐ Baja
Necesidad	☒ Esencial ☐ Deseable ☐ Opcional	**Verificabilidad**	☒ Alta ☐ Media ☐ Baja

Tabla 38: RF-023, Entrega de mensaje por el NAO

RF-024			
Nombre	Detección y reconocimiento de marca		
Descripción	El sistema podrá buscar una marca de las reconocibles por el NAO, detectando dicha marca al encontrarla y reconociendo de cuál se trata.		
Fecha	03/03/2019	**Estabilidad**	☒ Alta ☐ Media ☐ Baja
Prioridad	☒ Alta ☐ Media ☐ Baja	**Claridad**	☒ Alta ☐ Media ☐ Baja
Necesidad	☒ Esencial ☐ Deseable ☐ Opcional	**Verificabilidad**	☒ Alta ☐ Media ☐ Baja

Tabla 39: RF-024, Detección y reconocimiento de marca

RF-025			
Nombre	Pérdida de marca		
Descripción	El sistema podrá volver a encontrar una marca reconocible por el NAO en caso de que este la pierda de vista.		
Fecha	03/03/2019	**Estabilidad**	☒ Alta ☐ Media ☐ Baja
Prioridad	☒ Alta ☐ Media ☐ Baja	**Claridad**	☒ Alta ☐ Media ☐ Baja
Necesidad	☒ Esencial ☐ Deseable ☐ Opcional	**Verificabilidad**	☒ Alta ☐ Media ☐ Baja

Tabla 40: RF-025, Pérdida de marca

RF-026			
Nombre	Caída del NAO		
Descripción	El sistema podrá hacer que el NAO se levante del suelo en caso de una caída y vuelva a buscar la marca que estaba buscando o alcanzando.		
Fecha	03/03/2019	**Estabilidad**	☒ Alta ☐ Media ☐ Baja
Prioridad	☒ Alta ☐ Media ☐ Baja	**Claridad**	☒ Alta ☐ Media ☐ Baja
Necesidad	☒ Esencial ☐ Deseable ☐ Opcional	**Verificabilidad**	☒ Alta ☐ Media ☐ Baja

Tabla 41: RF-026, Caída del NAO

RF-027			
Nombre	Alcance de marca		
Descripción	El sistema podrá hacer que el NAO se acerque físicamente a una marca reconocible hasta estar a cuarenta centímetros de distancia de ella.		
Fecha	03/03/2019	**Estabilidad**	☒ Alta ☐ Media ☐ Baja
	☒ Alta		☒ Alta

Prioridad	☐ Media ☐ Baja	**Claridad**	☐ Media ☐ Baja
Necesidad	☒ Esencial ☐ Deseable ☐ Opcional	**Verificabilidad**	☒ Alta ☐ Media ☐ Baja

Tabla 42: RF-027, Alcance de marca

RF-028			
Nombre	Alcance de última marca		
Descripción	El sistema podrá hacer que el NAO empiece a buscar personas cuando haya alcanzado la última marca de la ruta que debía recorrer.		
Fecha	03/03/2019	**Estabilidad**	☒ Alta ☐ Media ☐ Baja
Prioridad	☒ Alta ☐ Media ☐ Baja	**Claridad**	☒ Alta ☐ Media ☐ Baja
Necesidad	☒ Esencial ☐ Deseable ☐ Opcional	**Verificabilidad**	☒ Alta ☐ Media ☐ Baja

Tabla 43: RF-028, Alcance de última marca

RF-029			
Nombre	Búsqueda de persona		
Descripción	El sistema podrá buscar a una persona, detectándola al encontrarla.		
Fecha	03/03/2019	**Estabilidad**	☒ Alta ☐ Media ☐ Baja
Prioridad	☒ Alta ☐ Media ☐ Baja	**Claridad**	☒ Alta ☐ Media ☐ Baja
Necesidad	☒ Esencial ☐ Deseable ☐ Opcional	**Verificabilidad**	☒ Alta ☐ Media ☐ Baja

Tabla 44: RF-029, Búsqueda de persona

RF-030			
Nombre	Persona encontrada incorrecta		
Descripción	El sistema podrá hacer que el NAO busque de nuevo a una persona en caso de que encuentre a una que no es el usuario receptor.		
Fecha	03/03/2019	**Estabilidad**	☒ Alta ☐ Media ☐ Baja
Prioridad	☒ Alta ☐ Media ☐ Baja	**Claridad**	☒ Alta ☐ Media ☐ Baja
Necesidad	☒ Esencial ☐ Deseable ☐ Opcional	**Verificabilidad**	☒ Alta ☐ Media ☐ Baja

Tabla 45: RF-030, Persona encontrada incorrecta

RF-031			
Nombre	Reproducción de mensaje		
Descripción	El sistema podrá reproducir el audio que contiene un mensaje si el NAO encuentra al usuario receptor de dicho mensaje.		
Fecha	03/03/2019	**Estabilidad**	☒ Alta ☐ Media ☐ Baja
Prioridad	☒ Alta ☐ Media ☐ Baja	**Claridad**	☒ Alta ☐ Media ☐ Baja
Necesidad	☒ Esencial ☐ Deseable ☐ Opcional	**Verificabilidad**	☒ Alta ☐ Media ☐ Baja

Tabla 46: RF-031, Reproducción de mensaje

RF-032			
Nombre	Respuesta de mensaje		
Descripción	El sistema podrá dar la opción al usuario receptor de contestar al mensaje en caso de haber recibido el mensaje personalmente por parte del NAO.		
Fecha	03/03/2019	**Estabilidad**	☒ Alta ☐ Media ☐ Baja
	☒ Alta		☒ Alta

Prioridad	☐ Media ☐ Baja	**Claridad**	☐ Media ☐ Baja
Necesidad	☒ Esencial ☐ Deseable ☐ Opcional	**Verificabilidad**	☒ Alta ☐ Media ☐ Baja

Tabla 47: RF-032, Respuesta de mensaje

RF-033			
Nombre	Reconstrucción de ruta		
Descripción	El sistema podrá reconstruir la ruta de marcas a la inversa que acaba de realizar el NAO con el objetivo de entregar una respuesta al usuario emisor inicial, siempre que dicha respuesta no sea personal.		
Fecha	03/03/2019	**Estabilidad**	☒ Alta ☐ Media ☐ Baja
Prioridad	☒ Alta ☐ Media ☐ Baja	**Claridad**	☒ Alta ☐ Media ☐ Baja
Necesidad	☒ Esencial ☐ Deseable ☐ Opcional	**Verificabilidad**	☒ Alta ☐ Media ☐ Baja

Tabla 48: RF-033, Reconstrucción de ruta

RF-034			
Nombre	Salida del sistema		
Descripción	El sistema podrá permitir al usuario la opción de salir del sistema.		
Fecha	03/03/2019	**Estabilidad**	☒ Alta ☐ Media ☐ Baja
Prioridad	☒ Alta ☐ Media ☐ Baja	**Claridad**	☒ Alta ☐ Media ☐ Baja
Necesidad	☒ Esencial ☐ Deseable ☐ Opcional	**Verificabilidad**	☒ Alta ☐ Media ☐ Baja

Tabla 49: RF-034, Salida del sistema

4.4.2 Requisitos no funcionales

Como ya se ha explicado, los requisitos no funcionales definen las restricciones que el sistema debe cumplir. A continuación, se detalla la lista de requisitos no funcionales identificados:

RNF-001			
Nombre	Sistema operativo		
Descripción	El sistema operativo debe ser Linux de 64 bits.		
Fecha	03/03/2019	**Estabilidad**	☒ Alta ☐ Media ☐ Baja
Prioridad	☒ Alta ☐ Media ☐ Baja	**Claridad**	☒ Alta ☐ Media ☐ Baja
Necesidad	☒ Esencial ☐ Deseable ☐ Opcional	**Verificabilidad**	☒ Alta ☐ Media ☐ Baja

Tabla 50: RNF-001, Sistema operativo

RNF-002			
Nombre	Versión de Choregraphe		
Descripción	La versión de Choregraphe para ejecutar el sistema debe ser la versión 2.1.4 o posterior.		
Fecha	03/03/2019	**Estabilidad**	☒ Alta ☐ Media ☐ Baja
Prioridad	☒ Alta ☐ Media ☐ Baja	**Claridad**	☒ Alta ☐ Media ☐ Baja
Necesidad	☒ Esencial ☐ Deseable ☐ Opcional	**Verificabilidad**	☒ Alta ☐ Media ☐ Baja

Tabla 51: RNF-002, Versión de Choregraphe

RNF-003	
Nombre	Lenguaje y versión del servidor
Descripción	La versión de Python para poder lanzar el servidor que hace realizar las acciones de bajo nivel al NAO debe ser la versión 2.7 o posterior.

Fecha	03/03/2019	**Estabilidad**	☒ Alta ☐ Media ☐ Baja
Prioridad	☒ Alta ☐ Media ☐ Baja	**Claridad**	☒ Alta ☐ Media ☐ Baja
Necesidad	☒ Esencial ☐ Deseable ☐ Opcional	**Verificabilidad**	☒ Alta ☐ Media ☐ Baja

Tabla 52: RNF-003, Lenguaje y versión del servidor

RNF-004			
Nombre	Versión de Java		
Descripción	La versión del software JDK (Java Development Kit) para ejecutar el sistema debe ser la versión 10 o posterior.		
Fecha	03/03/2019	**Estabilidad**	☒ Alta ☐ Media ☐ Baja
Prioridad	☒ Alta ☐ Media ☐ Baja	**Claridad**	☒ Alta ☐ Media ☐ Baja
Necesidad	☒ Esencial ☐ Deseable ☐ Opcional	**Verificabilidad**	☒ Alta ☐ Media ☐ Baja

Tabla 53: RNF-004, Versión de Java

RNF-005			
Nombre	Versión de GnuPG		
Descripción	La versión de la herramienta de cifrado y firmas digitales GNU Privacy Guard para cifrar el fichero de usuarios autorizados debe ser la versión 2.2.4 o superior.		
Fecha	03/03/2019	**Estabilidad**	☒ Alta ☐ Media ☐ Baja
Prioridad	☒ Alta ☐ Media ☐ Baja	**Claridad**	☒ Alta ☐ Media ☐ Baja

Necesidad	☒ Esencial ☐ Deseable ☐ Opcional	**Verificabilidad**	☒ Alta ☐ Media ☐ Baja

Tabla 54: RNF-005, Versión de GnuPG

RNF-006			
Nombre	Formato del dominio y del problema		
Descripción	Los ficheros que contienen el dominio y el problema a resolver deben estar escritos en el lenguaje PDDL.		
Fecha	03/03/2019	**Estabilidad**	☒ Alta ☐ Media ☐ Baja
Prioridad	☒ Alta ☐ Media ☐ Baja	**Claridad**	☒ Alta ☐ Media ☐ Baja
Necesidad	☒ Esencial ☐ Deseable ☐ Opcional	**Verificabilidad**	☒ Alta ☐ Media ☐ Baja

Tabla 55: RNF-006, Formato del dominio y del problema

RNF-007			
Nombre	Formato de ficheros de traducciones		
Descripción	Tanto el formato del fichero que traduce las acciones de alto nivel a acciones de bajo nivel como el formato del fichero que interpreta el estado de las acciones de bajo nivel deben seguir el formato de PELEA.		
Fecha	03/03/2019	**Estabilidad**	☒ Alta ☐ Media ☐ Baja
Prioridad	☒ Alta ☐ Media ☐ Baja	**Claridad**	☒ Alta ☐ Media ☐ Baja
Necesidad	☒ Esencial ☐ Deseable ☐ Opcional	**Verificabilidad**	☒ Alta ☐ Media ☐ Baja

Tabla 56: RNF-007, Formato del fichero de traducciones

RNF-008			
Nombre	Cifrado del fichero de usuarios		
Descripción	El fichero de usuarios autorizados para utilizar el sistema deberá estar cifrado con la herramienta de cifrado y firmas digitales GNU Privacy Guard.		
Fecha	03/03/2019	**Estabilidad**	☒ Alta ☐ Media ☐ Baja
Prioridad	☒ Alta ☐ Media ☐ Baja	**Claridad**	☒ Alta ☐ Media ☐ Baja
Necesidad	☒ Esencial ☐ Deseable ☐ Opcional	**Verificabilidad**	☒ Alta ☐ Media ☐ Baja

Tabla 57: RNF-008, Cifrado del fichero de usuarios

RNF-009			
Nombre	Seguridad de uso del sistema		
Descripción	El sistema no se podrá utilizar si no hay ningún usuario guardado en el fichero de usuarios autorizados.		
Fecha	03/03/2019	**Estabilidad**	☒ Alta ☐ Media ☐ Baja
Prioridad	☒ Alta ☐ Media ☐ Baja	**Claridad**	☒ Alta ☐ Media ☐ Baja
Necesidad	☒ Esencial ☐ Deseable ☐ Opcional	**Verificabilidad**	☒ Alta ☐ Media ☐ Baja

Tabla 58: RNF-009, Seguridad de uso del sistema

RNF-010			
Nombre	Restricción de usuarios autorizados		
Descripción	El sistema sólo permitirá su uso a usuarios que estén guardados correctamente en el fichero de usuarios autorizados y que tengan guardado su rostro en la base de datos del NAO.		
Fecha	03/03/2019	**Estabilidad**	☒ Alta ☐ Media

			☐ Baja
Prioridad	☒ Alta ☐ Media ☐ Baja	**Claridad**	☒ Alta ☐ Media ☐ Baja
Necesidad	☒ Esencial ☐ Deseable ☐ Opcional	**Verificabilidad**	☒ Alta ☐ Media ☐ Baja

Tabla 59: RNF-010, Restricción de usuarios autorizados

RNF-011			
Nombre	Restricción de funciones por rol		
Descripción	El usuario administrador podrá realizar todas las funciones que permite el sistema, mientras que un usuario normal sólo podrá realizar las funciones de comunicarse y salir.		
Fecha	03/03/2019	**Estabilidad**	☒ Alta ☐ Media ☐ Baja
Prioridad	☒ Alta ☐ Media ☐ Baja	**Claridad**	☒ Alta ☐ Media ☐ Baja
Necesidad	☒ Esencial ☐ Deseable ☐ Opcional	**Verificabilidad**	☒ Alta ☐ Media ☐ Baja

Tabla 60: RNF-011, Restricción de funciones por rol

RNF-012			
Nombre	Duración del mensaje grabado		
Descripción	La duración del mensaje que grabe el sistema deberá de ser de diez, treinta o sesenta segundos.		
Fecha	03/03/2019	**Estabilidad**	☒ Alta ☐ Media ☐ Baja
Prioridad	☒ Alta ☐ Media ☐ Baja	**Claridad**	☒ Alta ☐ Media ☐ Baja
Necesidad	☒ Esencial ☐ Deseable ☐ Opcional	**Verificabilidad**	☒ Alta ☐ Media ☐ Baja

Tabla 61: RNF-012, Duración del mensaje grabado

4.5 Matriz de trazabilidad

Con el objetivo de asegurar que cada caso de uso queda cubierto por al menos un requisito funcional, se ha realizado la siguiente matriz de trazabilidad entre los requisitos funcionales y los casos de uso:

	CU-001	CU-002	CU-003	CU-004	CU-005	CU-006	CU-007	CU-008	CU-009	CU-010	CU-011	CU-012	CU-013
RF-001	X		X	X	X	X	X	X	X	X	X	X	X
RF-002			X	X	X	X	X	X	X	X	X	X	X
RF-003			X	X	X	X	X	X	X	X	X	X	X
RF-004	X		X	X	X	X	X	X	X	X	X	X	X
RF-005	X		X	X	X	X	X	X	X	X	X	X	X
RF-006		X	X	X	X	X	X	X	X	X	X	X	X
RF-007		X	X	X	X	X	X	X	X	X	X	X	X
RF-008		X	X	X	X	X	X	X	X	X	X	X	X
RF-009		X	X	X	X	X	X	X	X	X	X	X	X
RF-010			X	X	X	X	X	X	X	X	X	X	X
RF-011				X	X	X	X	X	X	X	X	X	X
RF-012				X									
RF-013					X	X	X	X					
RF-014					X								
RF-015						X							
RF-016							X						
RF-017								X					
RF-018									X	X	X	X	
RF-019									X	X	X	X	
RF-020									X	X	X	X	
RF-021									X		X		
RF-022									X		X		
RF-023									X	X	X	X	
RF-024										X	X	X	
RF-025										X	X	X	

	CU-001	CU-002	CU-003	CU-004	CU-005	CU-006	CU-007	CU-008	CU-009	CU-010	CU-011	CU-012	CU-013
RF-026										X	X	X	
RF-027										X	X	X	
RF-028										X	X	X	
RF-029										X	X	X	
RF-030										X	X	X	
RF-031										X	X	X	
RF-032											X	X	
RF-033												X	
RF-034													X

Tabla 62: Matriz de trazabilidad

5. DISEÑO E IMPLEMENTACIÓN DEL SISTEMA

El objetivo de este capítulo es mostrar el diseño y la implementación del sistema que ha sido descrito y modelado en el capítulo de análisis. Para ello, se realizará una descripción de la arquitectura del sistema, se explicarán los pasos seguidos en su implementación y se expondrán las diferentes alternativas de solución que han ido surgiendo durante la realización del trabajo.

5.1 Arquitectura del sistema

En el inicio del diseño del sistema, se partía de la arquitectura de PELEA explicada en el capítulo del estado del arte:

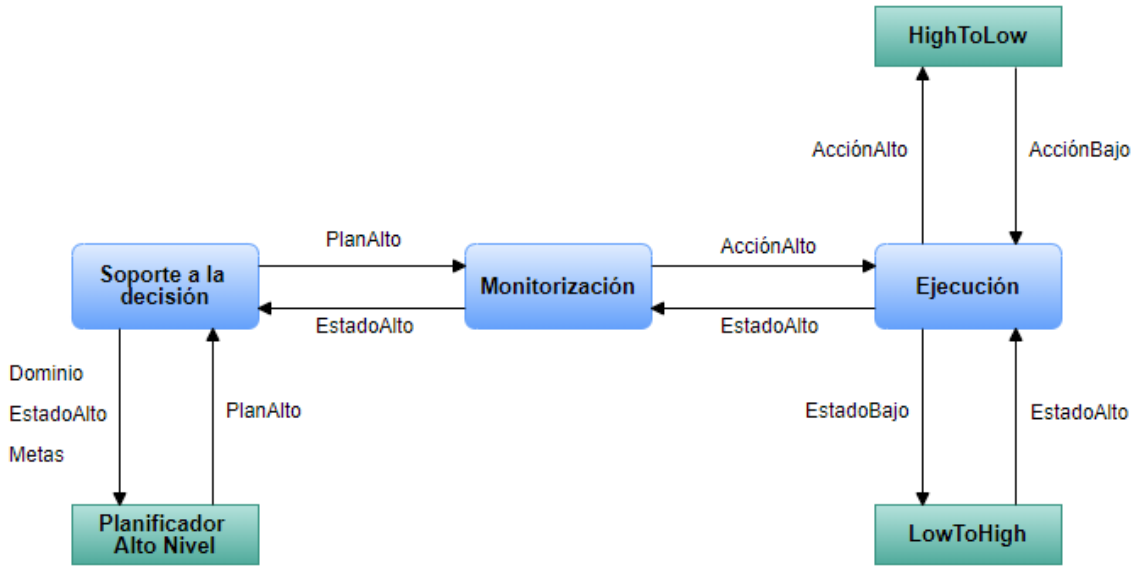


Ilustración 26: Arquitectura de PELEA

Las consideraciones que se realizaron sobre la versión de PELEA de la que se disponía consistieron en decidir qué elementos había que modificar y qué elementos había que añadir a dicha arquitectura.

Con respecto a la parte de los elementos que había que modificar de PELEA, se tuvo que realizar una nueva especificación de dos ficheros: *HighToLow.cfg* y *LowToHigh.cfg*. Estos ficheros son con los que trabajan los componentes *HighToLow* (*HighToLow.cfg*) y *LowToHigh* (*LowToHigh.cfg*), y es donde se realiza la traducción de una acción de alto nivel a una acción de bajo nivel (*HighToLow.cfg*) y la traducción de un estado de bajo nivel a un estado de alto nivel (*LowToHigh.cfg*), respectivamente.

Además, se tuvo que sustituir el dominio y el problema que venía escrito con PDDL en PELEA por el problema modelado para este trabajo.

Con respecto a la parte de los elementos que le faltaban a PELEA, fundamentalmente la parte que no estaba implementada era la comunicación que debía existir entre la arquitectura PELEA y el robot NAO.

Para realizar el diseño de esta parte, se pensó en la arquitectura cliente-servidor [31]. En ella, el cliente realiza la petición de un servicio a un servidor, que es el encargado de realizar dicho servicio, y le devuelve posteriormente al cliente la respuesta oportuna:

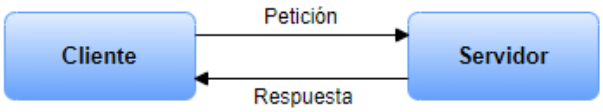


Ilustración 27: Arquitectura cliente-servidor

En la arquitectura cliente servidor, el servidor está continuamente escuchando peticiones de clientes, devolviendo para cada petición una respuesta, y la comunicación entre el cliente y el servidor se realiza mediante TCP/IP.

Una vez que se tenía claro el funcionamiento de la arquitectura cliente-servidor, se debía pensar cómo acoplar esta arquitectura con la arquitectura de PELEA.

La solución a esta integración se basó en la siguiente idea:

- **Cliente:** dado que es el que realiza las peticiones al servidor, en este caso el cliente será PELEA y las peticiones consistirán en cada una de las acciones de bajo nivel que son mandadas al robot NAO.
- **Servidor:** teniendo en cuenta que no se dispone de un servidor propio específicamente, que las acciones que se mandan al NAO están escritas en Python y que PELEA es una arquitectura que está programada en Java, la solución pasaba por implementar un servidor desde cero. Este servidor estaría programado en Python y se encargaría de recibir una acción de bajo nivel de PELEA (cliente), realizar dicha acción en el NAO y mandar una respuesta a PELEA que contenga el estado de bajo nivel en el que se encuentra el problema.

El estado de la arquitectura, tras la implementación del servidor, sería el siguiente:

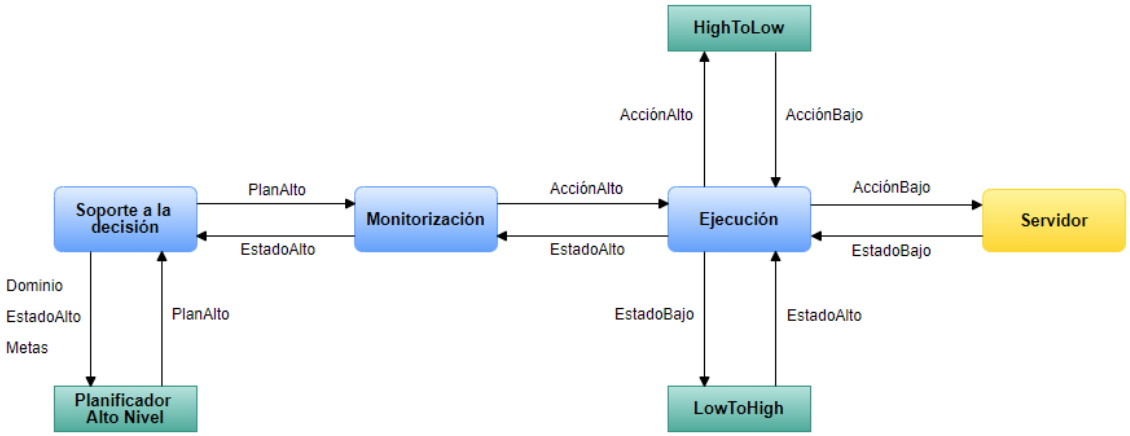


Ilustración 28: Arquitectura intermedia del sistema

Además, como se comentó en el capítulo de análisis del sistema, con la idea de aumentar la seguridad del sistema se creó un fichero que contiene en cada una de sus líneas el nombre y los apellidos de un usuario, su correo electrónico y su rol.

De esta forma, el servidor se encarga de leer dicho fichero con el objetivo de consultar si un usuario está autorizado o no, obtener su nombre y su correo electrónico y consultar el rol de usuario que tenga.

Entonces, finalmente, la arquitectura del sistema quedaría de la siguiente forma:

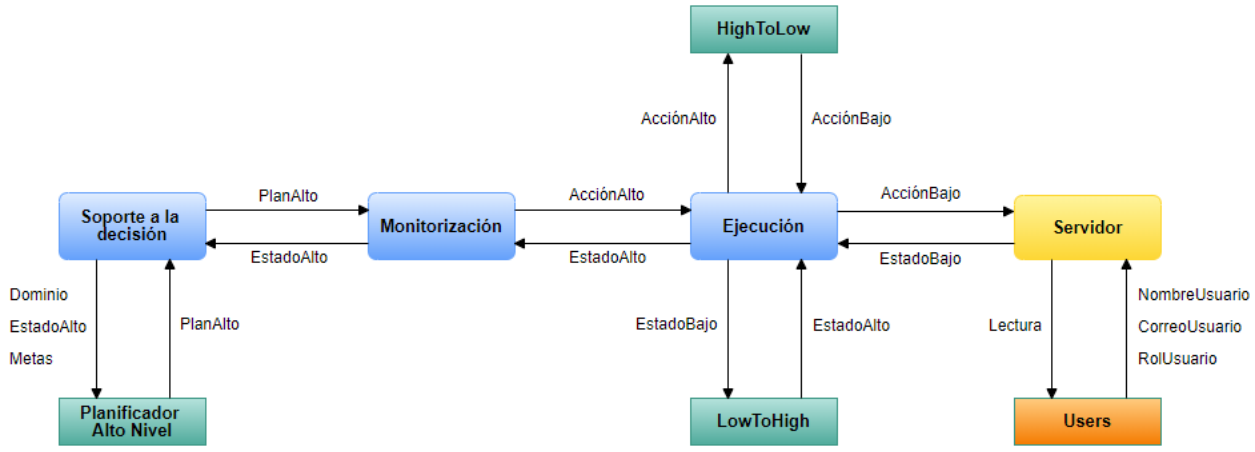


Ilustración 29: Arquitectura final del sistema

5.2 Modificaciones realizadas sobre PELEA

Los ficheros que van a trabajar conjuntamente durante la ejecución de PELEA son: *domainMessages.pddl*, *problemMessages.pddl*, *HighToLow.cfg* y *LowToHigh.cfg*. Estos ficheros son, respectivamente, el fichero que contiene el dominio en PDDL, el fichero que contiene el problema en PDDL, el fichero que traduce una acción de alto nivel a una acción de bajo nivel y el fichero que traduce un estado de bajo nivel a un estado de alto nivel.

El fichero *HighToLow.cfg* es el más sencillo de todos, pues se basa únicamente en la declaración de parejas de acciones (la de alto nivel con su pareja de bajo nivel). Así, dado que en este trabajo se tienen diecisiete acciones de alto nivel, entonces se tendrán las correspondientes diecisiete acciones de bajo nivel. Hay que reseñar que una acción de alto nivel podría tener que descomponerse en varias acciones de bajo nivel, aunque en el caso de este trabajo no ha sido necesario.

Por tanto, la representación de la acción de alto nivel en el fichero *HighToLow.cfg* es exactamente igual a su representación en PDDL, así, por ejemplo, para la acción *getFunction* su representación en alto y bajo nivel (en ese orden específico) es el siguiente:

```
High: getFunction(function, robot)
Lows: getFunctionLowLevel()
```

Ilustración 30: Acción de alto y bajo nivel en HighToLow

Como se puede observar, la variación entre la acción de alto nivel y la acción de bajo nivel consiste en la adición del sufijo *LowLevel* al nombre de la acción de alto nivel para formar el nombre de la acción de bajo nivel y en la ausencia de parámetros en la acción de bajo nivel.

Esta ausencia de parámetros se da en todas las parejas de acciones menos en dos parejas concretas:

```
High: findMark(function, robot, message, wayPoint1, wayPoint2)
Lows: findMarkLowLevel($wayPoint2)

High: goToMark(function, robot, message, wayPoint1, wayPoint2)
Lows: goToMarkLowLevel($wayPoint2)
```

Ilustración 31: Excepción en HighToLow

Esta diferencia se da porque en las acciones *findMarkLowLevel* y *goToMarkLowLevel* es necesario tener como parámetro la marca que se quiere buscar (*findMark*) o la marca a la que se quiere llegar (*goToMark*).

Como se puede comprobar, la implementación se ha realizado de tal forma que los parámetros de la acción de bajo nivel deben llevar un $ seguido del nombre del parámetro en la acción de alto nivel.

Una vez que se entendió como funcionaba PDDL, era necesario modelar el problema a resolver con este lenguaje de planificación, lo que implicaba la creación de un dominio y la creación de un problema sobre la situación que se pretende resolver.

En primer lugar, se va a realizar la explicación del dominio. En el dominio, la primera sección que se tiene es la sección de los tipos de objeto:

```
(:types
    function
    objective
    robot
    message
    person
    wayPoint
)
```

Ilustración 32: Tipos de objeto en el dominio

En esta sección se tienen los siguientes tipos de objeto: *function* (utilizado para especificar la función del sistema que se quiere utilizar), *objective* (utilizado para controlar si el objetivo del problema se ha alcanzado o no), *robot* (utilizado para representar al robot NAO), *message* (utilizado para representar los estados por los que se encontrará el mensaje a entregar), *person* (utilizado para representar a la persona a la que hay que entregar el mensaje), y *wayPoint* (utilizado para representar cada una de las posiciones o marcas del mapa que el NAO debe seguir para entregar un mensaje a un usuario receptor).

La segunda sección del dominio es la de los predicados. Esta sección, dada su gran longitud, va a ser explicada por grupos de predicados.

El primer grupo de predicados es el siguiente:

```
(undefined ?f - function)
(learnFace ?f - function)
(searchFace ?f - function)
(listFaces ?f - function)
(removeFace ?f - function)
(clearDatabase ?f - function)
(communicate ?f - function)
```

Ilustración 33: Primer grupo de predicados del dominio

En él, se tienen los predicados que representan cada una de las funciones que el sistema puede realizar: *undefined* (valor inicial de la función), *learnFace* (valor de la función cuando la función elegida es la de aprender un nuevo rostro), *searchFace* (valor de la función cuando la función elegida es la de buscar un rostro), *listFaces* (valor de la función cuando la función elegida es la de listar todos los rostros), *removeFace* (valor de la función cuando la función elegida es la de eliminar un rostro), *clearDatabase* (valor de la función cuando la función elegida es la de eliminar todos los rostros) y *communicate* (valor de la función cuando la función elegida es la de enviar un mensaje a un usuario receptor).

El segundo grupo de predicados es el siguiente:

```
(achieved ?o - objective)
(available ?r - robot)
```

Ilustración 34: Segundo grupo de predicados del dominio

En él, se tienen los siguientes predicados: *achieved* (para representar que el objetivo del problema se ha alcanzado) y *available* (para representar que el robot NAO está disponible).

El tercer grupo de predicados es el siguiente:

```
(init ?m - message)
(sender ?m - message)
(recorded ?m - message)
(send ?m - message)
(navigate ?m - message)
(received ?m - message)
```

Ilustración 35: Tercer grupo de predicados del dominio

En él, se tienen los predicados que representan cada uno de los estados por los que pasa el mensaje en el problema: *init* (estado inicial del mensaje), *sender* (estado del mensaje cuando ya se conoce al usuario receptor que recibirá dicho mensaje), *recorded* (estado del mensaje cuando este ya ha sido grabado), *send* (estado del mensaje cuando va a ser entregado por correo), *navigate* (estado del mensaje cuando va a ser entregado por el propio NAO) y *received* (estado del mensaje cuando ha sido entregado al usuario receptor por el propio NAO y este decide responder a dicho mensaje).

El cuarto grupo de predicados es el siguiente:

```
(search ?p - person)
(unknown ?p - person)
(known ?p - person)
```

Ilustración 36: Cuarto grupo de predicados del dominio

En él, se tienen los estados por los que pasa la persona que se encuentra el NAO cuando va buscando al usuario receptor: *search* (estado de la persona cuando el NAO ha alcanzado la última marca de su ruta), *unknown* (estado de la persona cuando el NAO detecta a una persona), *known* (estado de la persona cuando el NAO la reconoce y se trata del usuario receptor).

El quinto grupo de predicados es el siguiente:

```
(origin ?x - wayPoint)
(stay ?r - robot ?x - wayPoint)
(can-travel ?r - robot ?x - wayPoint)
(connected ?x - wayPoint ?y - wayPoint)
(stand ?p - person ?x - wayPoint)
```

Ilustración 37: Quinto grupo de predicados del dominio

En él se tienen los siguientes predicados: *origin* (para indicar la marca desde la que parte el NAO en la ruta que sigue), *stay* (para indicar que el NAO se encuentra en una determinada marca), *can-travel* (para indicar que el NAO puede moverse a una determinada marca), *connected* (para indicar que dos marcas están conectadas y se puede ir desde la primera hasta la segunda) y *stand* (para indicar que el usuario receptor se encuentra en una determinada marca).

La tercera sección del dominio es la de las posibles acciones a realizar. Este problema tiene diecisiete posibles acciones a realizar, que van a ser explicadas a continuación.

Antes, es necesario especificar que el efecto de las acciones que está puesto en el fichero *domainMessages.pddl* se trata del efecto ideal de la acción. Cuando el módulo de *Ejecución* ejecuta una acción de alto nivel, aplica su efecto ideal al estado del problema y por tanto es un efecto que se realiza, obteniendo con ello el estado esperado del problema. Entonces, cuando el servidor devuelve el estado de la acción a bajo nivel, se comprueba las consecuencias de ese estado en el fichero *LowToHigh.cfg*. Por tanto, dependiendo del estado que la acción de bajo nivel ha devuelto, se ejecutarán comandos de adición (*pddl_add*) o eliminación (*pddl_delete*) de predicados sobre el estado esperado del problema, obteniendo con ello el estado real del problema. Evidentemente, si en el efecto ideal de la acción de alto nivel se añade un predicado y de la acción de bajo nivel se obtiene un estado en el que dicho predicado no puede existir, en el fichero *LowToHigh.cfg* habrá que añadir los comandos oportunos para eliminarlo. Una vez que el módulo de *Ejecución* obtiene el estado real del

problema, lo manda al módulo de *Monitorización* para que lo compare con el estado esperado y vea si son iguales o no.

La primera acción del dominio es ***getFunction*** y su estructura es la siguiente:

```
(:action getFunction
    :parameters (?f - function ?r - robot ?o - objective)
    :precondition (and (undefined ?f)(available ?r))
    :effect (and (not(undefined ?f))(communicate ?f))
)
```

Ilustración 38: Acción getFunction

Es la acción de alto nivel encargada de obtener la función que el usuario quiere realizar. Como precondiciones tiene que la función del problema sea indefinida y que el NAO esté disponible. Como efecto ideal tiene que la función deje de ser indefinida y pase a ser la función de comunicarse.

Esta acción, dependiendo del valor que devuelva el servidor en el parámetro *$function*, tendrá los siguientes efectos en el fichero *LowToHigh.cfg*:

```
If: $function is communicate
pddl_delete(undefined function)
pddl_add(communicate function)

If: $function is learn
pddl_delete(communicate function)
pddl_add(learnFace function)

If: $function is search
pddl_delete(communicate function)
pddl_add(searchFace function)

If: $function is list
pddl_delete(communicate function)
pddl_add(listFaces function)

If: $function is remove
pddl_delete(communicate function)
pddl_add(removeFace function)

If: $function is clear
pddl_delete(communicate function)
pddl_add(clearDatabase function)

If: $function is exit
pddl_add(achieved objective)

If: $function is denied
pddl_add(achieved objective)
```

Ilustración 39: Efectos de getFunction en LowToHigh

Como se puede observar, dependiendo del valor del parámetro recibido la acción tendrá un efecto distinto:

- *communicate*: la función deja de ser indefinida y pasa a ser la función de comunicarse.

- *learn*: se elimina el efecto ideal de que la función es comunicarse y la función pasa a ser la de aprender un rostro.
- *search*: se elimina el efecto ideal de que la función es comunicarse y la función pasa a ser la de buscar un rostro.
- *list*: se elimina el efecto ideal de que la función es comunicarse y la función pasa a ser la de listar todos los rostros.
- *remove*: se elimina el efecto ideal de que la función es comunicarse y la función pasa a ser la de eliminar un rostro.
- *clear*: se elimina el efecto ideal de que la función es comunicarse y la función pasa a ser la de eliminar todos los rostros.
- *exit*: cuando el usuario quiere salir, se da el objetivo del problema como alcanzado, para que así termine la ejecución del sistema.
- *denied*: cuando se deniega el acceso al usuario, se da el objetivo del problema como alcanzado, para que así termine la ejecución del sistema.

La segunda acción del dominio es ***learnFace*** y su estructura es la siguiente:

```
(:action learnFace
    :parameters (?f - function ?r - robot ?o - objective)
    :precondition (and (learnFace ?f)(available ?r))
    :effect (achieved ?o)
)
```

Ilustración 40: Acción learnFace

Es la acción de alto nivel encargada de guardar un rostro. Como precondiciones tiene que la función sea la de aprender un rostro y que el NAO esté disponible. Como efecto ideal tiene que el objetivo del problema pase a estar alcanzado.

Esta acción, dependiendo del valor que devuelva el servidor en el parámetro *$learn*, tendrá los siguientes efectos en el fichero *LowToHigh.cfg*:

```
If: $learn is true
pddl_add(achieved objective)

If: $learn is false
pddl_add(achieved objective)
```

Ilustración 41: Efectos de learnFace en LowToHigh

Como se puede observar, para los dos posibles valores del parámetro recibido por el servidor la acción tendrá el mismo efecto:

- *true*: si se ha guardado el rostro, se da el objetivo del problema como alcanzado, para que así termine la ejecución del sistema.
- *false*: si no se ha guardado el rostro, se da el objetivo del problema como alcanzado, para que así termine la ejecución del sistema.

Esto se ha implementado de esta manera para que tanto si el usuario administrador consigue guardar un rostro como si no, se termine la ejecución del sistema.

La tercera acción del dominio es ***searchFace*** y su estructura es la siguiente:

```
(:action searchFace
    :parameters (?f - function ?r - robot ?o - objective)
    :precondition (and (searchFace ?f)(available ?r))
    :effect (achieved ?o)
)
```

Ilustración 42: Acción searchFace

Es la acción de alto nivel encargada de buscar un rostro. Como precondiciones tiene que la función sea la de buscar un rostro y que el NAO esté disponible. Como efecto ideal tiene que el objetivo del problema pase a estar alcanzado.

Para los efectos de esta acción en el fichero *LowToHigh.cfg* dependiendo del valor del parámetro recibido por el servidor es igual que en el caso de la acción *learnFace*, sólo que esta vez el nombre del parámetro es *$search* y los valores *true* y *false* van referidos a si se ha encontrado un rostro o no.

La cuarta acción del dominio es ***listFaces*** y su estructura es la siguiente:

```
(:action listFaces
    :parameters (?f - function ?r - robot ?o - objective)
    :precondition (and (listFaces ?f)(available ?r))
    :effect (achieved ?o)
)
```

Ilustración 43: Acción listFaces

Es la acción de alto nivel encargada de listar todos los rostros. Como precondiciones tiene que la función sea la de listar todos los rostros y que el NAO esté disponible. Como efecto ideal tiene que el objetivo del problema pase a estar alcanzado.

Para los efectos de esta acción en el fichero *LowToHigh.cfg* dependiendo del valor del parámetro recibido por el servidor es igual que en el caso de la acción *learnFace*, sólo que esta vez el nombre del parámetro es *$list* y los valores *true* y *false* van referidos a si se han listado todos los rostros o no.

La quinta acción del dominio es ***removeFace*** y su estructura es la siguiente:

```
(:action removeFace
    :parameters (?f - function ?r - robot ?o - objective)
    :precondition (and (removeFace ?f)(available ?r))
    :effect (achieved ?o)
)
```

Ilustración 44: Acción removeFace

Es la acción de alto nivel encargada de eliminar un rostro. Como precondiciones tiene que la función sea la de eliminar un rostro y que el NAO esté disponible. Como efecto ideal tiene que el objetivo del problema pase a estar alcanzado.

Para los efectos de la acción en el fichero *LowToHigh.cfg* dependiendo del valor del parámetro recibido por el servidor es igual que en el caso de la acción *learnFace*, sólo que esta vez el nombre del parámetro es *$remove* y los valores *true* y *false* van referidos a si se ha eliminado un rostro o no.

La sexta acción del dominio es ***clearDatabase*** y su estructura es la siguiente:

```
(:action clearDatabase
    :parameters (?f - function ?r - robot ?o - objective)
    :precondition (and (clearDatabase ?f)(available ?r))
    :effect (achieved ?o)
)
```

Ilustración 45: Acción clearDatabase

Es la acción de alto nivel encargada de eliminar todos los rostros. Como precondiciones tiene que la función sea la de eliminar todos los rostros y que el NAO esté disponible. Como efecto ideal tiene que el objetivo del problema pase a estar alcanzado.

Para los efectos de la acción en el fichero *LowToHigh.cfg* dependiendo del valor del parámetro recibido por el servidor es igual que en el caso de la acción *learnFace*, sólo que esta vez el nombre del parámetro es *$clear* y los valores *true* y *false* van referidos a si se han eliminado todos los rostros o no.

La séptima acción del dominio es ***getInformation*** y su estructura es la siguiente:

```
(:action getInformation
    :parameters (?f - function ?r - robot ?m - message)
    :precondition (and (communicate ?f)(available ?r)(init ?m))
    :effect (and (not(init ?m))(sender ?m))
)
```

Ilustración 46: Acción getInformation

Es la acción de alto nivel encargada de obtener los datos sobre el usuario receptor. Como precondiciones tiene que la función sea la de comunicarse, que el NAO esté disponible y que el mensaje se encuentre en su estado inicial. Como efecto ideal tiene que el mensaje deje de estar en su estado inicial para que pase al estado en el que se conoce al usuario receptor del mensaje.

Esta acción, dependiendo del valor que devuelva el servidor en el parámetro *$sender*, tendrá los siguientes efectos en el fichero *LowToHigh.cfg*:

```
If: $sender is true
pddl_delete(init message0)
pddl_add(sender message0)

If: $sender is false
pddl_add(achieved objective)
```

Ilustración 47: Efectos de getInformation en LowToHigh

Como se puede observar, dependiendo del valor del parámetro recibido la acción tendrá un efecto distinto:

- *true*: se ha identificado al usuario receptor, por lo que el mensaje deja de estar en su estado inicial para que pase al estado en el que se conoce al usuario receptor del mensaje.
- *false*: ha habido problemas en la identificación del usuario receptor, por lo que se da el objetivo del problema como alcanzado, para que así termine la ejecución del sistema.

La octava acción del dominio es ***recordMessage*** y su estructura es la siguiente:

```
(:action recordMessage
    :parameters (?f - function ?r - robot ?m - message)
    :precondition (and (communicate ?f)(available ?r)(sender ?m))
    :effect (and (not(sender ?m))(recorded ?m))
)
```

Ilustración 48: Acción recordMessage

Es la acción de alto nivel encargada de grabar el mensaje del usuario emisor. Como precondiciones tiene que la función sea la de comunicarse, que el NAO esté disponible y que el mensaje se encuentre en el estado en el que se conoce al usuario receptor del mensaje. Como efecto ideal tiene que el mensaje deje de estar en el estado en el que se conoce al usuario receptor para que pase al estado en el que se encuentra grabado.

Esta acción, dependiendo del valor que devuelva el servidor en el parámetro *$record*, tendrá los siguientes efectos en el fichero *LowToHigh.cfg*:

```
If: $record is true
pddl_delete(sender message0)
pddl_add(recorded message0)

If: $record is false
//Not contemplated.
```

Ilustración 49: Efectos de recordMessage en LowToHigh

Como se puede observar, dependiendo del valor del parámetro recibido la acción tendrá un efecto distinto:

- *true*: se ha grabado el mensaje, por lo que el mensaje deja de estar en el estado en el que se conoce al usuario receptor para que pase al estado en el que se encuentra grabado.
- *false*: este caso no se contempla, dado que se trata de una acción que a bajo nivel no tiene complejidad.

La novena acción del dominio es ***typeMessage*** y su estructura es la siguiente:

```
(:action typeMessage
    :parameters (?f - function ?r - robot ?m - message)
    :precondition (and (communicate ?f)(available ?r)(recorded ?m))
    :effect (and (not(recorded ?m))(send ?m))
)
```

Ilustración 50: Acción typeMessage

Es la acción de alto nivel encargada de obtener la vía por la que se quiere entregar el mensaje al usuario receptor, ya sea mandándolo por correo o entregándolo el propio NAO. Como precondiciones tiene que la función sea la de comunicarse, que el NAO esté disponible y que el mensaje se encuentre grabado. Como efecto ideal tiene que el mensaje pase del estado en el que se encuentra grabado al estado en el que se manda por correo.

Esta acción, dependiendo del valor que devuelva el servidor en el parámetro *$mode*, tendrá los siguientes efectos en el fichero *LowToHigh.cfg*:

```
If: $mode is email
pddl_delete(recorded message0)
pddl_add(send message0)

If: $mode is route
pddl_delete(send message0)
pddl_add(navigate message0)

If: $mode is incorrect
pddl_add(achieved objective)
```

Ilustración 51: Efectos de typeMessage en LowToHigh

Como se puede observar, dependiendo del valor del parámetro recibido la acción tendrá un efecto distinto:

- *email*: el mensaje se va a mandar por correo, por lo que se elimina el estado por el cual el mensaje está grabado y se añade el estado de que el mensaje va a ser mandado por correo.
- *route*: el mensaje lo va a entregar el propio NAO, por lo que se elimina el estado por el cual el mensaje se manda por correo y se añade el estado de que el mensaje será entregado por el propio NAO.
- *incorrect*: el mensaje lo va a entregar el propio NAO, pero el usuario receptor no tiene su rostro almacenado, por lo que se da el objetivo del problema como alcanzado, para que así termine la ejecución del sistema.

La décima acción del dominio es ***sendMessage*** y su estructura es la siguiente:

```
(:action sendMessage
    :parameters (?f - function ?r - robot ?m - message ?o - objective)
    :precondition (and (communicate ?f)(available ?r)(send ?m))
    :effect (achieved ?o)
)
```

Ilustración 52: Acción sendMessage

Es la acción de alto nivel encargada de enviar el mensaje por correo al usuario receptor. Como precondiciones tiene que la función sea la de comunicarse, que el NAO esté disponible y que el mensaje vaya a ser entregado por correo. Como efecto ideal tiene que el objetivo del problema pase a estar alcanzado.

Esta acción, dependiendo del valor que devuelva el servidor en el parámetro *$send*, tendrá los siguientes efectos en el fichero *LowToHigh.cfg*:

```
If: $send is true
pddl_add(achieved objective)

If: $send is false
//Not contemplated.
```

Ilustración 53: Efectos de sendMessage en LowToHigh

Como se puede observar, dependiendo del valor del parámetro recibido la acción tendrá un efecto distinto:

- *true*: el mensaje se ha mandado por correo, por lo que se da el objetivo del problema como alcanzado, para que así termine la ejecución del sistema.
- *false*: este caso no se ha contemplado, dado que no se puede comprobar que el mensaje enviado llega al correo electrónico del usuario receptor.

La undécima acción del dominio es ***findMark*** y su estructura es la siguiente:

```
(:action findMark
    :parameters (?f - function ?r - robot ?m - message ?x - wayPoint ?y - wayPoint)
    :precondition (and (communicate ?f)(available ?r)(navigate ?m)(stay ?r ?x)
    (connected ?x ?y))
    :effect (can-travel ?r ?y)
)
```

Ilustración 54: Acción findMark

Es la acción de alto nivel encargada de buscar una marca concreta que el NAO reconoce. Como precondiciones tiene que la función sea la de comunicarse, que el NAO esté disponible, que el mensaje vaya a ser entregado por el propio NAO, que el NAO se encuentre en una determinada marca y que la marca en la que se encuentra el NAO y la siguiente marca estén conectadas. Como efecto ideal tiene que se añada el predicado que permite al NAO viajar hasta la siguiente marca de la ruta.

Esta acción, dependiendo del valor que devuelva el servidor en el parámetro *$found*, tendrá los siguientes efectos en el fichero *LowToHigh.cfg*:

```
If: $found is true
pddl_add(can-travel robot0 parameter1)

If: $found is false
//Not contemplated.
```

Ilustración 55: Efectos de findMark en LowToHigh

Como se puede observar, dependiendo del valor del parámetro recibido la acción tendrá un efecto distinto:

- *true*: el NAO ha encontrado la marca que estaba buscando, por lo que se añade el predicado que permite al NAO viajar hasta la siguiente marca de la ruta.
- *false*: este caso no se ha contemplado, dado que el NAO en esta acción va rotando sobre sí mismo hasta que encuentra la marca que está buscando.

La duodécima acción del dominio es ***goToMark*** y su estructura es la siguiente:

```
(:action goToMark
    :parameters (?f - function ?r - robot ?m - message ?x - wayPoint ?y - wayPoint)
    :precondition (and (communicate ?f)(available ?r)(navigate ?m)(stay ?r ?x)
    (can-travel ?r ?y))
    :effect (and (not(stay ?r ?x))(not(can-travel ?r ?y))(stay ?r ?y))
)
```

Ilustración 56: Acción goToMark

Es la acción de alto nivel encargada de acercar al NAO hasta la marca que ha localizado. Como precondiciones tiene que la función sea la de comunicarse, que el NAO esté disponible, que el mensaje vaya a ser entregado por el propio NAO, que el NAO se encuentre en una determinada marca y que desde esa marca el NAO pueda viajar a la siguiente marca. Como efecto ideal tiene que se elimine la marca antigua donde estaba el NAO, se elimine el predicado que le ha permitido llegar hasta la nueva marca y se añada la marca nueva donde se encuentra ahora el NAO.

Esta acción, dependiendo del valor que devuelva el servidor en el parámetro *$destination*, tendrá los siguientes efectos en el fichero *LowToHigh.cfg*:

```
If: $destination is true
pddl_delete(stay robot0 parameter0)
pddl_delete(can-travel robot0 parameter1)
pddl_add(stay robot0 parameter1)

If: $destination is false
pddl_delete(stay robot0 parameter1)
pddl_add(stay robot0 parameter0)
```

Ilustración 57: Efectos de goToMark en LowToHigh

Como se puede observar, dependiendo del valor del parámetro recibido la acción tendrá un efecto distinto:

- *true*: el NAO ha llegado hasta la marca, por lo que se elimina la marca antigua donde estaba el NAO, se elimina el predicado que le ha permitido llegar hasta la nueva marca y se añade la marca nueva donde se encuentra ahora el NAO.
- *false*: el NAO ha perdido de vista la marca o se ha caído al suelo, por lo que se elimina al NAO de la nueva marca que había alcanzado y se le devuelve a su antigua marca.

La decimotercera acción del dominio es ***checkGoalMark*** y su estructura es la siguiente:

```
(:action checkGoalMark
    :parameters (?f - function ?r - robot ?m - message ?x - wayPoint ?p - person)
    :precondition (and (communicate ?f)(available ?r)(navigate ?m)(stay ?r ?x)
    (stand ?p ?x))
    :effect (search ?p)
)
```

Ilustración 58: Acción checkGoalMark

Es la acción de alto nivel encargada de comprobar que el NAO y el usuario receptor se encuentran en la misma marca. Como precondiciones tiene que la función sea la de comunicarse, que el NAO esté disponible, que el mensaje vaya a ser entregado por el propio NAO, que el NAO se encuentre en una determinada marca y que el usuario receptor se encuentre en la misma marca que el NAO. Como efecto ideal tiene que se añada el predicado de buscar al usuario receptor.

Esta acción, dependiendo del valor que devuelva el servidor en el parámetro *$check*, tendrá los siguientes efectos en el fichero *LowToHigh.cfg*:

```
If: $check is true
pddl_add(search person0)

If: $check is false
//Not contemplated.
```

Ilustración 59: Efectos de checkGoalMark en LowToHigh

Como se puede observar, dependiendo del valor del parámetro recibido la acción tendrá un efecto distinto:

- *true*: se ha llegado a la última marca de la ruta, por lo que se añade el predicado de buscar al usuario receptor.

- *false*: este caso no se contempla, dado que se trata de una acción que a bajo nivel no tiene complejidad.

La decimocuarta acción del dominio es ***findPerson*** y su estructura es la siguiente:

```
(:action findPerson
    :parameters (?f - function ?r - robot ?m - message ?p - person)
    :precondition (and (communicate ?f)(available ?r)(navigate ?m)
    (search ?p))
    :effect (and (not(search ?p))(unknown ?p))
)
```

Ilustración 60: Acción findPerson

Es la acción de alto nivel encargada de buscar a una persona. Como precondiciones tiene que la función sea la de comunicarse, que el NAO esté disponible, que el mensaje vaya a ser entregado por el propio NAO y que se deba buscar al usuario receptor. Como efecto ideal tiene que se elimine el predicado de buscar al usuario receptor y se añada el predicado de persona desconocida.

Esta acción, dependiendo del valor que devuelva el servidor en el parámetro *$person*, tendrá los siguientes efectos en el fichero *LowToHigh.cfg*:

```
If: $person is true
pddl_delete(search person0)
pddl_add(unknown person0)

If: $person is false
pddl_delete(unknown person0)
pddl_add(search person0)
```

Ilustración 61: Efectos de findPerson en LowToHigh

Como se puede observar, dependiendo del valor del parámetro recibido la acción tendrá un efecto distinto:

- *true*: se ha encontrado a una persona, por lo que se elimina el predicado de buscar al usuario receptor y se añade el predicado de persona desconocida.
- *false*: no se ha encontrado a ninguna persona, por lo que se elimina el predicado de persona desconocida y se añade el predicado de buscar al usuario receptor, con el objetivo de que se vuelva a realizar esta acción.

La decimoquinta acción del dominio es ***recognisePerson*** y su estructura es la siguiente:

```
(:action recognisePerson
    :parameters (?f - function ?r - robot ?m - message ?p - person)
    :precondition (and (communicate ?f)(available ?r)(navigate ?m)
    (unknown ?p))
    :effect (and (not(unknown ?p))(known ?p))
)
```

Ilustración 62: Acción recognisePerson

Es la acción de alto nivel encargada de reconocer al usuario receptor. Como precondiciones tiene que la función sea la de comunicarse, que el NAO esté disponible, que el mensaje vaya a ser entregado por el propio NAO y que la persona sea desconocida. Como efecto ideal tiene que se elimine el predicado de persona desconocida y se añada el predicado de usuario receptor reconocido.

Esta acción, dependiendo del valor que devuelva el servidor en el parámetro *$recognise*, tendrá los siguientes efectos en el fichero *LowToHigh.cfg*:

```
If: $recognise is true
pddl_delete(unknown person0)
pddl_add(known person0)

If: $recognise is false
pddl_delete(known person0)
pddl_add(search person0)
```

Ilustración 63: Efectos de recognisePerson en LowToHigh

Como se puede observar, dependiendo del valor del parámetro recibido la acción tendrá un efecto distinto:

- *true*: se ha reconocido a la persona y es el usuario receptor, por lo que se elimina el predicado de persona desconocida y se añade el predicado de usuario receptor reconocido.
- *false*: no se ha reconocido a la persona o se la ha reconocido, pero no es el usuario receptor, por lo que se elimina el predicado de usuario receptor reconocido y se añade el predicado de buscar al usuario receptor, con el objetivo de que se vuelva a realizar la acción *findPerson*.

La decimosexta acción del dominio es ***playMessage*** y su estructura es la siguiente:

```
(:action playMessage
    :parameters (?f - function ?r - robot ?m - message ?p - person
    ?o - objective)
    :precondition (and (communicate ?f)(available ?r)(navigate ?m)
    (known ?p))
    :effect (and (not(navigate ?m))(received ?m))
)
```

Ilustración 64: Acción playMessage

Es la acción de alto nivel encargada de reproducir el mensaje grabado al usuario receptor. Como precondiciones tiene que la función sea la de comunicarse, que el NAO esté disponible, que el mensaje vaya a ser entregado por el propio NAO y que se haya encontrado al usuario receptor. Como efecto ideal tiene que el objetivo del problema pase a estar alcanzado.

Esta acción, dependiendo del valor que devuelva el servidor en el parámetro *$answer*, tendrá los siguientes efectos en el fichero *LowToHigh.cfg*:

```
If: $answer is true
pddl_delete(navigate message0)
pddl_add(received message0)

If: $answer is false
pddl_add(achieved objective)
```

Ilustración 65: Efectos de playMessage en LowToHigh

Como se puede observar, dependiendo del valor del parámetro recibido la acción tendrá un efecto distinto:

- *true*: el usuario receptor quiere responder al mensaje que ha recibido, por lo que el mensaje deja de estar en el estado en el que va a ser entregado por el propio NAO para que pase al estado de mensaje recibido que va a ser respondido.
- *false*: el usuario receptor no quiere responder al mensaje, por lo que se da el objetivo del problema como alcanzado, para que así termine la ejecución del sistema.

La decimoséptima y última acción del dominio es ***rebuildRoute*** y su estructura es la siguiente:

```
(:action rebuildRoute
    :parameters (?f - function ?r - robot ?m - message ?x - wayPoint
    ?y - wayPoint ?p - person)
    :precondition (and (communicate ?f)(available ?r)(received ?m)
    (stay ?r ?x)(stand ?p ?x)(origin ?y)(known ?p))
    :effect (and (sender ?m)(stand ?p ?y)(origin ?x)(not(received ?m))
    (not(stand ?p ?x))(not(origin ?y))(not(known ?p)))
)
```

Ilustración 66: Acción rebuildRoute

Es la acción de alto nivel encargada de reconstruir el estado para que el NAO devuelva al usuario emisor la respuesta al mensaje que ha recibido el usuario receptor. Como precondiciones tiene que la función sea la de comunicarse, que el NAO esté disponible, que el mensaje haya sido recibido y vaya a ser respondido, que el NAO se encuentre en una determinada marca, que el usuario receptor se encuentre en la misma marca que el NAO, que se tenga el predicado que guarda la marca origen desde la que partió el NAO y que el usuario receptor haya sido reconocido. Como efecto ideal tiene que se añada el estado del mensaje en el que se conoce el usuario receptor del mensaje, que el usuario receptor se encuentre ahora en la marca de la que partió el NAO y que la marca origen de la que parte el NAO ahora sea la marca actual en la que se encuentra, eliminando el estado de mensaje recibido por el usuario receptor y que va a ser respondido, la antigua marca en la que se encontraba el usuario receptor, la marca origen antigua de la que partió el NAO y el predicado que indica que el usuario receptor ha sido reconocido.

En definitiva, lo que pretende realizar esta acción es invertir el problema del que se viene, haciendo que el usuario receptor este en la primera marca de la ruta que ha hecho el NAO (es decir, el que antes era el usuario emisor ahora es el usuario receptor), y que a partir de la realización de esta acción se realice de nuevo la acción *recordMessage* para repetir todo el proceso, sólo que esta vez el usuario que interactúa con el NAO es el usuario receptor del mensaje (que ahora es el usuario emisor del mensaje que se envía como respuesta).

Por tanto, a esta acción sólo se puede llegar en el caso de que el mensaje que grabó el usuario emisor haya sido entregado por el propio NAO y no por correo electrónico.

Esta acción, dependiendo del valor que devuelva el servidor en el parámetro *$rebuild*, tendrá los siguientes efectos:

```
If: $rebuild is true
pddl_delete(received message0)
pddl_delete(navigateWay way)
pddl_delete(at-stand person0 parameter0)
pddl_delete(origin parameter1)
pddl_delete(known person0)
pddl_add(sender message0)
pddl_add(sendWay way)
pddl_add(at-stand person0 parameter1)
pddl_add(origin parameter0)

If: $rebuild is false
//Not contemplated.
```

Ilustración 67: Efectos de rebuildRoute en LowToHigh

Como se puede observar, dependiendo del valor del parámetro recibido la acción tendrá un efecto distinto:

- *true*: se realizan todas las adiciones y eliminaciones que se habían realizado en el efecto ideal de la acción.
- *false*: este caso no se contempla, dado que se trata de una acción que a bajo nivel no tiene complejidad.

Por tanto, en resumen, las acciones que se realizan durante la ejecución del sistema tanto de alto nivel (color azul) como de bajo nivel (color verde), son las siguientes:

1) getFunction - getFunctionLowLevel
2) learnFace - learnFaceLowLevel
3) searchFace - searchFaceLowLevel
4) listFaces - listFacesLowLevel
5) removeFace - removeFaceLowLevel
6) clearDatabase - clearDatabaseLowLevel
7) getInformation - getInformationLowLevel
8) recordMessage - recordMessageLowLevel
9) typeMessage - typeMessageLowLevel
10) sendMessage - sendMessageLowLevel
11) findMark - findMarkLowLevel
12) goToMark - goToMarkLowLevel
13) checkGoalMark - checkGoalMarkLowLevel
14) findPerson - findPersonLowLevel
15) recognisePerson - recognisePersonLowLevel
16) playMessage - playMessageLowLevel
17) rebuildRoute - rebuildRouteLowLevel

Ilustración 68: Acciones del sistema

Con respecto al fichero del problema en PDDL, este tiene tres secciones importantes a explicar.

La primera sección es donde se declaran los objetos que se van a utilizar en el problema:

```
(:objects
    function - function
    objective - objective
    robot0 - robot
    message0 - message
    person0 - person
    naomark84 - wayPoint
    naomark85 - wayPoint
    naomark107 - wayPoint
)
```

Ilustración 69: Objetos del problema

Como se puede ver, se declara un objeto de tipo *function* (para utilizarlo en la especificación de la función del sistema), un objeto de tipo *objective* (para utilizarlo para la definición de la meta), un objeto de tipo *robot* (para representar al NAO en el problema), un objeto de tipo *message* (para representar al mensaje que se va a entregar), un objeto de tipo *person* (para representar al usuario receptor) y, en este caso, tres objetos de tipo *waypoint* (para representar las marcas que se van a utilizar en el problema).

La segunda sección es donde se declaran los predicados de los que parte el problema:

```
(:init
    (undefined function)
    (available robot0)
    (init message0)
    (origin naomark84)
    (stay robot0 naomark84)
    (connected naomark84 naomark85)
    (connected naomark85 naomark107)
    (connected naomark107 naomark85)
    (connected naomark85 naomark84)
    (stand person0 naomark107)
)
```

Ilustración 70: Predicados del problema

Como se puede observar, se declara la función como indefinida, se pone al NAO como disponible, se asigna al mensaje a su estado inicial, se especifica la marca desde la que parte el NAO, se coloca al NAO en dicha marca, se realizan las conexiones entre las marcas del problema y se define la marca donde el usuario receptor va a estar. Con respecto a la posible variedad de problemas, los predicados que son fijos son los tres primeros, siendo los predicados restantes dependientes del problema en cuestión que se quiera resolver, como viene explicado en el documento anexo de manual de instalación.

La tercera y última sección es donde se especifica la meta a alcanzar en el problema:

```
(:goal
    (achieved objective)
)
```

Ilustración 71: Meta del problema

En esta sección se define que el problema habrá sido resuelto cuando el objetivo del problema haya sido alcanzado.

Teniendo en cuenta las funciones que permite el sistema, en la siguiente tabla se muestran las acciones que se realizarían durante la ejecución de una función concreta (aunque se ponga sólo el nombre de las acciones de alto nivel, como ya se ha explicado, también se realizarían sus correspondientes versiones a bajo nivel):

Función	Acciones
Aprender un rostro	*getFunction* y *learnFace*
Buscar un rostro	*getFunction* y *searchFace*
Listar todos los rostros	*getFunction* y *listFaces*
Eliminar un rostro	*getFunction* y *removeFace*
Eliminar todos los rostros	*getFunction* y *clearDatabase*
Entrega de mensaje por correo	*getFunction*, *getInformation*, *recordMessage*, *typeMessage* y *sendMessage*
Entrega de mensaje por el propio NAO	*getFunction*, *getInformation*, *recordMessage*, *typeMessage*, *findMark*, *goToMark*, *checkGoalMark*, *findPerson*, *recognisePerson*, *playMessage* y *rebuildRoute**(Si el usuario quiere responder al mensaje recibido)
Salir	*getFunction*

Tabla 63: Funciones del sistema

Se debe recordar, como ya se dejó detallado en el análisis del sistema, que el único usuario que puede realizar todas las funciones descritas en la tabla es el usuario administrador, mientras un usuario que tenga un rol normal sólo puede realizar la función de entregar un mensaje por correo o por el propio NAO y la función de salir del sistema.

Por último, a parte de la creación del dominio y del problema en PDDL y de la modificación de los ficheros de traducciones, en el módulo de *Ejecución* también se añadió la parte de comunicación con el servidor de Python.

Para ello, utilizando el lenguaje Java, se empleó un *socket* para establecer la comunicación con el servidor. Para el intercambio de información, se utilizó por un lado las clases *OutputStream* y *DataOutputSream* para mandar una petición al servidor y, por otro lado, las clases *InputStream* y *DataInputStream* para recoger la respuesta del servidor.

5.3 Configuración del fichero de usuarios

Como se especificó en el análisis del sistema, el servidor utilizará un fichero llamado *Users.cfg* para poder realizar:

- La identificación de los usuarios autorizados (los que están dentro de este fichero),
- La distinción dentro del grupo de usuarios autorizados entre el usuario administrador y los usuarios normales (evaluando el rol de usuario).
- La obtención de los datos de los usuarios que los identifican (nombre y correo).

Teniendo esto en cuenta, un ejemplo correcto de formato del fichero *Users.cfg* es:

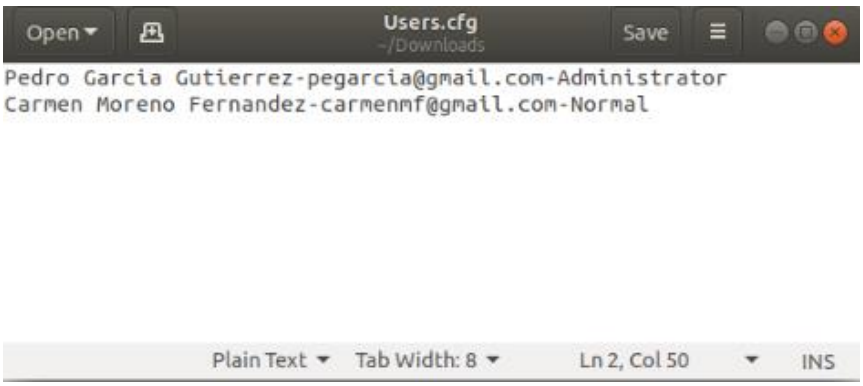


Ilustración 72: Fichero de usuarios autorizados

En este fichero, cada línea está compuesta por: el nombre y los apellidos del usuario, un delimitador (-), su correo electrónico, un delimitador (-) y su rol en el sistema:

- *Administrator*: se trata del usuario administrador.
- *Normal*: se trata de un usuario normal.

Para proteger este fichero de su consulta o modificación, se utilizó la herramienta de cifrado y firmas digitales GnuPG, obteniendo con ello el fichero *Users.cfg.gpg*, que es el fichero *Users.cfg* cifrado.

5.4 Módulos del NAO utilizados

En el momento de empezar a crear scripts para realizar acciones en el NAO, este robot tiene organizadas sus distintas funcionalidades en módulos, cada uno de los cuales posee una API donde están los métodos que permiten utilizar todas las posibilidades del propio módulo.

De esta forma, de todos los módulos que tiene el robot NAO, se han utilizado los siguientes en la implementación de este trabajo:

- *ALTextToSpeech* [2]: es el módulo que permite al NAO hablar.
- *ALSpeechRecognition* [3]: es el módulo que permite al NAO reconocer palabras de un idioma.
- *ALMemory* [4]: es el módulo que actúa como una memoria centralizada para guardar información importante para el NAO.
- *ALAudioRecorder* [5]: es el módulo que permite al NAO grabar audios de voz.
- *ALRobotPosture* [6]: es el módulo que permite controlar las posturas predefinidas del NAO.
- *ALMotion* [7]: es el módulo que permite el movimiento del NAO.
- *ALTracker* [8]: es el módulo que permite al NAO localizar visualmente a un determinado objetivo.
- *ALPeoplePerception* [9]: es el módulo que permite al NAO detectar a las personas.
- *ALFaceDetection* [10]: es el módulo que permite al NAO memorizar y eliminar rostros de personas, además de poder detectar una cara y reconocerla en caso de que la tuviera memorizada previamente.
- *ALAudioPlayer* [11]: es el módulo que permite al NAO reproducir audios de voz.

A continuación, se procede a explicar los métodos utilizados para el trabajo de cada uno de los módulos explicados.

Del primer módulo, ***ALTextToSpeech***, el único método que se ha utilizado es *say*, al que se le pasa como argumento el texto escrito que se quiere que el NAO diga.

Del segundo módulo, ***ALSpeechRecognition***, se han utilizado los siguientes métodos: *setLanguage* (permite elegir el idioma en el que se va a realizar el reconocimiento de palabras), *setVocabulary* (permite establecer el grupo de palabras del idioma elegido que el NAO debe reconocer), *subscribe* (permite suscribirse al evento de reconocer palabras) y *unsubscribe* (permite cancelar la subscripción al evento de reconocer palabras). Por tanto, el reconocimiento de palabras se realiza después de haberse suscrito al evento de reconocer

palabras y antes de cancelar la subscripción en él. Para obtener aquellas palabras que el NAO ha reconocido, se debe utilizar el módulo *ALMemory* y su método *getData* para acceder a la clave de memoria *WordRecognized*, que es donde están guardadas las palabras.

Del tercer módulo, ***ALMemory***, se han utilizado únicamente dos métodos: *getData* y *removeData*. El método *getData* permite obtener de una determinada clave de memoria los datos almacenados en ella y el método *removeData* permite eliminar los datos almacenados en una determinada clave de memoria.

Del cuarto módulo, ***ALAudioRecorder***, se han utilizado dos métodos: *startMicrophonesRecording* y *stopMicrophonesRecording*. El método *startMicrophonesRecording* permite que el NAO empiece a grabar un audio y el método *stopMicrophonesRecording* permite que el NAO deje de grabar dicho audio.

Del quinto módulo, ***ALRobotPosture***, se han utilizado dos métodos: *getPosture* y *goToPosture*. El método *getPosture* permite obtener la postura en la que se encuentra el NAO y el método *goToPosture* permite que el NAO se ponga en una determinada postura.

Del sexto módulo, ***ALMotion***, se han utilizado tres métodos: *moveInit* (inicializa el proceso de moverse), *moveTo* (permite al NAO moverse con una determinada distancia y un determinado ángulo de rotación) y *rest* (permite al NAO relajar todas sus articulaciones).

Del séptimo módulo, ***ALTracker***, se han utilizado seis métodos: *registerTarget* (permite registrar un objetivo a rastrear especificando los detalles de este, como su tamaño o el tipo de objetivo), *track* (permite iniciar el rastreo del objetivo especificando el tipo de objetivo que se va a rastrear, que en este trabajo serán las marcas que el NAO reconoce), *getTargetPosition* (permite obtener la distancia vectorizada del objetivo a rastrear), *isTargetLost* (permite comprobar si se ha perdido de vista el objetivo), *stopTracker* (permite detener el rastreo del objetivo) y *unregisterAllTargets* (permite borrar todos los objetivos a rastrear registrados). Con respecto a los tipos de objetivo que el NAO puede rastrear, a parte de las marcas que se utilizan en este trabajo, también se pueden rastrear pelotas rojas, personas, caras de persona o sonidos. Con respecto al rastreo de objetivos con el NAO, durante este rastreo se puede realizar la búsqueda de varios objetivos y no sólo uno. Por ejemplo, se pueden registrar dos marcas concretas y que el NAO empiece a buscar ambas.

Del octavo módulo, ***ALPeoplePerception***, se han utilizado dos métodos: *subscribe* (permite suscribirse al evento de detectar personas) y *unsubscribe* (permite cancelar la subscripción al

evento de detectar personas). Al igual que ocurría en el reconocimiento de palabras, la detección de personas se realiza después de haberse suscrito al evento de detectar personas y antes de cancelar la subscripción en él. Para saber si el NAO ha detectado a alguna persona, se comprueba si en la clave de memoria *PeoplePerception/PeopleDetected* hay algún valor mediante el módulo *ALMemory* y el método *getData*. Si hay algún valor almacenado, es que el NAO ha detectado alguna persona y si está vacío, es que no ha detectado a ninguna.

Del noveno módulo, ***ALFaceDetection***, se han utilizado seis métodos: *learnFace* (permite, a partir de un nombre como parámetro, guardar de forma vectorizada en la base de datos del NAO el rostro que el NAO tiene delante de su cara), *getLearnedFacesList* (permite obtener la lista de nombres, es decir, rostros, que el NAO tiene almacenados en su base de datos), *forgetPerson* (permite, a partir de un nombre como parámetro, borrar el rostro de una persona de la base de datos del NAO), *clearDatabase* (permite borrar todos los rostros almacenados en la base de datos del NAO), *subscribe* (permite suscribirse al evento de detectar caras), *unsubscribe* (permite cancelar la subscripción al evento de detectar caras). Al igual que ocurría en el reconocimiento de palabras, la detección de caras se realiza después de haberse suscrito al evento de detectar caras y antes de cancelar la subscripción en él. Para obtener datos sobre la cara de la persona detectada, se accede a la clave de memoria *FaceDetected* con el módulo *ALMemory* y el método *getData*. Como puntualización, se aclara que cuando una cara es guardada en la base de datos del NAO, lo que realmente se guarda de ella es el nombre de la persona y los detalles de la cara vectorizados, pero no se guarda ninguna imagen de la cara de la persona. Cuando se utiliza el término vectorizada, se refiere a que se guardan de forma numérica los rasgos de la cara de la persona (por ejemplo, de la nariz se guarda el tamaño y su posición).

Del décimo módulo, ***ALAudioPlayer***, se han utilizado dos métodos: *loadFile* (permite cargar el fichero que contiene el audio grabado) y *play* (permite reproducir el audio grabado).

5.5 Implementación del servidor

Al igual que en el apartado 5.2 se explicó el funcionamiento de las diecisiete acciones en alto nivel, con la explicación de la implementación del servidor se va a describir el contenido de estas acciones a bajo nivel, con el objetivo de que las acciones queden totalmente definidas tanto en el aspecto de alto nivel como en el de bajo nivel.

La acción ***getFunctionLowLevel*** es la acción de bajo nivel encargada de obtener la función que el usuario quiere realizar. En primer lugar, desencripta el fichero *Users.cfg.gpg* para

obtener los usuarios autorizados para utilizar el sistema utilizando la librería *gnupg*. En caso de que no haya usuarios autorizados guardados la acción devolverá en el parámetro *function* el valor *denied*. Si hay usuarios guardados, se pide al usuario que coloque su rostro delante de la cara del NAO para iniciar el reconocimiento de rostro. En caso de que el usuario no sea reconocido, se devuelve en el parámetro *function* el valor *denied*. Si el usuario es reconocido, se comprueba que esté en la lista de usuarios autorizados. En caso de que el usuario no sea un usuario autorizado, se devuelve en el parámetro *function* el valor *denied*. Si el usuario es autorizado, se comprueba que su correo está escrito y que tiene un formato correcto, devolviendo en el parámetro *function* el valor *denied* en caso de que no sea así. Si todas las comprobaciones se superan, se da al usuario la opción de escuchar o no la explicación de las funciones que tiene disponibles. Si es un usuario administrador y contesta que sí, se le explican todas las funciones del sistema para que elija una de ellas y si dice que no simplemente se le pide que elija una de ellas. En el caso del usuario normal, el comportamiento es igual con la excepción de que en vez de utilizar todas las funciones, sólo se permiten las funciones de comunicarse y salir. Dependiendo de la función escogida por el usuario, la acción devolverá en el parámetro *function* un valor distinto: *learn* (aprender un rostro), *search* (buscar un rostro), *list* (listar todos los rostros), *remove* (eliminar un rostro), *clear* (eliminar todos los rostros), *communicate* (comunicarse con otro usuario) y *exit* (salir).

Los módulos del NAO utilizados en esta acción son: *ALTextToSpeech* (para comunicar frases al usuario), *ALFaceDetection* (para el reconocimiento de rostro del usuario), *ALSpeechRecognition* (para el reconocimiento de palabras en la contestación sobre si se quiere explicación de las funciones o no y en la elección de la función que se quiere realizar) y *ALMemory* (para acceder a las claves de memoria *FaceDetected* y *WordRecognized*).

La acción ***learnFaceLowLevel*** es la acción de bajo nivel encargada de guardar un rostro en la base de datos del NAO. En primer lugar, vuelve a desencriptar el fichero *Users.cfg.gpg* para obtener la lista de usuarios autorizados y obtiene también la lista de rostros (nombres) guardados en la base de datos del NAO. Entonces se pide al usuario que diga el nombre del usuario del que quiere guardar el rostro y se comprueba que es un usuario autorizado. En caso de que no sea un usuario autorizado, se devuelve en el parámetro *learn* el valor *false*. Si es un usuario autorizado, se comprueba que no tenga ya guardado el rostro en la base de datos del NAO. En caso de que ya lo tenga guardado, se devuelve en el parámetro *learn* el valor *false*. Si no lo tiene ya guardado, se pide al usuario que ponga el rostro del usuario que quiere

guardar delante de la cara del NAO, se guarda su nombre y su rostro en la base de datos del NAO y se devuelve en el parámetro *learn* el valor *true*.

Los módulos del NAO utilizados en esta acción son: *ALTextToSpeech* (para comunicar frases al usuario), *ALSpeechRecognition* (para reconocer el nombre del usuario del que se quiere guardar el rostro), *ALMemory* (para acceder a las claves de memoria *FaceDetected* y *WordRecognized*) y *ALFaceDetection* (para la detección de rostro, para obtener la lista de rostros guardados y para guardar el rostro).

La acción ***searchFaceLowLevel*** es la acción de bajo nivel encargada de buscar si un rostro está guardado o no en la base de datos del NAO. En primer lugar, obtiene la lista de rostros (nombres) guardados en la base de datos del NAO. Si esta lista no tiene ningún rostro (nombre), se devuelve en el parámetro *search* el valor *false*. Si tiene algún rostro, se pide al usuario que diga el nombre del usuario del que quiere saber si tiene el rostro guardado. Si el nombre dicho por el usuario está en la lista, el rostro está guardado en la base de datos del NAO y se devuelve en el parámetro *search* el valor *true*. En caso de que el rostro no esté guardado, se devuelve en el parámetro *search* el valor *false*.

Los módulos del NAO utilizados en esta acción son: *ALTextToSpeech* (para comunicar frases al usuario), *ALSpeechRecognition* (para reconocer el nombre del usuario del que se quiere buscar el rostro), *ALMemory* (para acceder a la clave de memoria *WordRecognized*) y *ALFaceDetection* (para obtener la lista de rostros guardados).

La acción ***listFacesLowLevel*** es la acción de bajo nivel encargada de mostrar todos los rostros guardados en la base de datos del NAO. En primer lugar, obtiene la lista de rostros (nombres) guardados en la base de datos del NAO. Si esta lista no tiene ningún rostro (nombre), se devuelve en el parámetro *list* el valor *false*. Si tiene algún rostro, se imprime la consola en la que se está ejecutando el servidor la lista de rostros (nombres) guardados y se devuelve en el parámetro *list* el valor *true*.

Los módulos del NAO utilizados en esta acción son: *ALTextToSpeech* (para comunicar frases al usuario) y *ALFaceDetection* (para obtener la lista de rostros guardados).

La acción ***removeFaceLowLevel*** es la acción de bajo nivel encargada de eliminar un rostro de la base de datos del NAO. En primer lugar, obtiene la lista de rostros (nombres) guardados en la base de datos del NAO. Si esta lista no tiene ningún rostro (nombre), se devuelve en el parámetro *remove* el valor *false*. Si tiene algún rostro, se pide al usuario que diga el nombre

del usuario del que se quiere eliminar el rostro. En caso de que el usuario del que se quiera eliminar el rostro no tenga el rostro guardado, se devuelve en el parámetro *remove* el valor *false*. En caso contrario, se elimina el rostro (nombre) de la base de datos del NAO y se devuelve en el parámetro *remove* el valor *true*.

Los módulos del NAO utilizados en esta acción son: *ALTextToSpeech* (para comunicar frases al usuario), *ALSpeechRecognition* (para reconocer el nombre del usuario del que se quiere eliminar el rostro), *ALMemory* (para acceder a la clave de memoria *WordRecognized*) y *ALFaceDetection* (para obtener la lista de rostros guardados y para eliminar el rostro).

La acción ***clearDatabaseLowLevel*** es la acción de bajo nivel encargada de eliminar todos los rostros de la base de datos del NAO. En primer lugar, obtiene la lista de rostros (nombres) guardados en la base de datos del NAO. Si esta lista no tiene ningún rostro (nombre), se devuelve en el parámetro *clear* el valor *false*. Si tiene algún rostro, se borran todos los rostros (nombres) de la base de datos del NAO y se devuelve en el parámetro *clear* el valor *true*.

Los módulos del NAO utilizados en esta acción son: *ALTextToSpeech* (para comunicar frases al usuario) y *ALFaceDetection* (para obtener la lista de rostros guardados y para eliminar todos los rostros).

La acción ***getInformationLowLevel*** es la acción de bajo nivel encargada de obtener la información requerida sobre el usuario receptor. En primer lugar, vuelve a desencriptar el fichero *Users.cfg.gpg* para obtener la lista de usuarios autorizados y obtiene también la lista de rostros (nombres) guardados en la base de datos del NAO. Entonces se pide al usuario que diga el nombre del usuario receptor y se comprueba que es un usuario autorizado. En caso de que no sea un usuario autorizado, se devuelve en el parámetro *sender* el valor *false*. Si es un usuario autorizado, se guarda en una variable si tiene el rostro guardado y se comprueba que su correo está escrito y que tiene un formato correcto, devolviendo en el parámetro *sender* el valor *false* en caso de que no sea así. Si ambas comprobaciones se superan, se guarda su nombre y su correo y se devuelve en el parámetro *sender* el valor *true*.

Los módulos del NAO utilizados en esta acción son: *ALTextToSpeech* (para comunicar frases al usuario), *ALSpeechRecognition* (para reconocer el nombre del usuario receptor), *ALMemory* (para acceder a la clave de memoria *WordRecognized*) y *ALFaceDetection* (para obtener la lista de rostros guardados).

La acción ***recordMessageLowLevel*** es la acción de bajo nivel encargada de grabar el mensaje del usuario emisor. En primer lugar, pregunta al usuario que opción de duración quiere que tenga su mensaje (diez, treinta o sesenta segundos). En función de la duración escogida, se graba el mensaje durante dicha duración y se devuelve en el parámetro *record* el valor *true*.

Los módulos del NAO utilizados en esta acción son: *ALTextToSpeech* (para comunicar frases al usuario), *ALSpeechRecognition* (para reconocer la duración del mensaje), *ALMemory* (para acceder a la clave de memoria *WordRecognized*) y *ALAudioRecorder* (para grabar el mensaje).

La acción ***typeMessageLowLevel*** es la acción de bajo nivel encargada de obtener la vía por la que se entrega el mensaje, ya sea por correo o por el propio NAO. En primer lugar, pregunta al usuario si el mensaje es personal o no lo es. Si el mensaje es personal, este será entregado por correo y se devuelve en el parámetro *mode* el valor *email*. Si el mensaje no es personal y el usuario receptor tiene rostro, el mensaje será entregado por el propio NAO y se devuelve en el parámetro *mode* el valor *route*. Finalmente, si el mensaje no es personal pero el usuario receptor no tiene el rostro almacenado, no se envía el mensaje y se devuelve en el parámetro *mode* el valor *incorrect*.

Los módulos del NAO utilizados en esta acción son: *ALTextToSpeech* (para comunicar frases al usuario), *ALSpeechRecognition* (para conocer si el mensaje es personal o no lo es) y *ALMemory* (para acceder a la clave de memoria *WordRecognized*).

La acción ***sendMessageLowLevel*** es la acción de bajo nivel encargada de mandar el audio grabado por correo al usuario receptor. En primer lugar, crea el mensaje a mandar especificando el correo desde donde se manda (*From*), que es el correo del NAO, el asunto del mensaje (*Subject*) y el cuerpo del mensaje utilizando las librerías *MIMEMultipart* y *MIMEText*. Una vez realizado esto, se añade al mensaje el correo donde va a ser enviado el mensaje (*To*). Dependiendo de cuál de los dos usuarios se trate, el correo donde se mande el mensaje será distinto. A continuación, se utiliza el protocolo *File Transfer Protocol* (*FTP*) para acceder al sistema de ficheros del NAO y obtener el fichero grabado que contiene el mensaje. Una vez obtenido, se adjunta al mensaje mediante la librería *MIMEAudio*. Cuando el mensaje está preparado, se utiliza la librería *smtplib* para conectarse con el servidor de Gmail, acceder a la cuenta de correo del NAO y enviar el mensaje. Para el envío de correos, se creó una cuenta de Gmail que se utilizará únicamente en este trabajo, y es a lo que se refiere "correo del NAO". Una vez enviado el mensaje, se devuelve en el parámetro *send* el

valor *true*, se coloca al NAO en la posición de sentado y se borran los valores de memoria necesarios.

Los módulos del NAO utilizados en esta acción son: *ALTextToSpeech* (para comunicar frases al usuario), *ALRobotPosture* (para colocar al robot en la posición de sentado), *ALTracker* (para detener el posible rastreo de marcas y borrar todos los objetivos a rastrear) y *ALMemory* (para borrar los datos almacenados en la clave de memoria *PeoplePerception/PeopleDetected*, ya que si no se realiza este borrado puede dar problemas una futura detección de personas).

La acción ***findMarkLowLevel*** es la acción de bajo nivel encargada de buscar una marca que el NAO reconoce. En primer lugar, coloca al NAO de pie. Después, registra la marca recibida por parámetro como el objetivo a rastrear y comienza el rastreo de dicha marca con la cámara de la cabeza del NAO. Si no ve ninguna marca, el NAO realiza una pequeña rotación sobre sí mismo hacia la izquierda y comprueba si ve la marca. Este proceso se realiza hasta que se encuentra la marca buscada y al final de cada movimiento de rotación se comprueba si el robot se ha caído, en cuyo caso se le vuelve a poner de pie y se continúa la búsqueda. Una vez encontrada la marca buscada, se devuelve en el parámetro *found* el valor *true*.

Los módulos del NAO utilizados en esta acción son: *ALRobotPosture* (para conocer la postura en la que está el NAO y ponerle de pie si fuera necesario), *ALTracker* (para registrar la marca a rastrear, comenzar su rastreo y comprobar si se está viendo), *ALMotion* (para realizar la rotación del NAO) y *ALTextToSpeech* (para comunicar cuando el NAO se ha caído o ha encontrado la marca).

La acción ***goToMarkLowLevel*** es la acción de bajo nivel encargada de acercar al NAO hasta la marca encontrada. En primer lugar, obtiene la distancia a la que se encuentra la marca. Una vez conocida dicha distancia, el NAO va avanzando hasta ella decímetro a decímetro controlando que la marca se encuentre siempre centrada. Si la marca está demasiado a la derecha, se rota hacia la derecha hasta que esté centrada y si está demasiado a la izquierda viceversa. Una vez corregida la colocación de la marca, se continúa avanzando hacia delante. Además, tanto en los avances rectos como en las rotaciones se va controlando que el NAO no se ha caído ni ha perdido de vista la marca, en cuyo caso se devolverá en el parámetro *destination* el valor *false* para que se vuelva a buscar la marca. Una vez que el NAO se encuentra aproximadamente a cuarenta centímetros de la marca, se detiene y se devuelve en el parámetro *destination* el valor *true*.

Los módulos del NAO utilizados en esta acción son: *ALTracker* (para obtener la distancia a la marca y controlar si la marca se ha perdido de vista), *ALRobotPosture* (para conocer la postura en la que está el NAO y ponerle de pie si fuera necesario), *ALMotion* (para realizar la rotación del NAO y los movimientos hacia delante) y *ALTextToSpeech* (para comunicar cuando el NAO se ha caído, ha perdido de vista la marca o ha alcanzado la marca).

La acción ***checkGoalMarkLowLevel*** es la acción de bajo nivel encargada de hacer comunicar al NAO que se ha alcanzado la última marca de la ruta. Comunica que se ha alcanzado la última marca de la ruta y devuelve en el parámetro *check* el valor *true*.

El único módulo utilizado en esta acción es *ALTextToSpeech* (para comunicar que se ha alcanzado la última marca de la ruta).

La acción ***findPersonLowLevel*** es la acción de bajo nivel encargada de encontrar a una persona. En primer lugar, comprueba que el NAO no se ha caído, levantándolo en caso de que así sea. Después, comprueba si en la cámara de la cabeza del NAO hay alguna persona. Si no hay ninguna persona, se realiza una pequeña rotación del NAO hacia la izquierda y se devuelve en el parámetro *person* el valor *false*. Si hay una persona, se devuelve en el parámetro *person* el valor *true*.

Los módulos del NAO utilizados en esta acción son: *ALTextToSpeech* (para comunicar cuando el NAO se ha caído, no ha encontrado a ninguna persona o ha encontrado a una persona), *ALRobotPosture* (para conocer la postura en la que está el NAO y ponerle de pie si fuera necesario), *ALPeoplePerception* (para controlar si se ve a una persona o no) *ALMemory* (para acceder a la clave de memoria *PeoplePerception/PeopleDetected* y borrar sus datos) y *ALMotion* (para realizar la rotación del NAO).

La acción ***recognisePersonLowLevel*** es la acción de bajo nivel encargada de reconocer al usuario receptor del mensaje. En primer lugar, pide la persona encontrada que ponga su rostro delante de la cara del NAO. Si el NAO no reconoce a la persona, se realiza una pequeña rotación del NAO hacia la izquierda y se devuelve en el parámetro *recognise* el valor *false*. Si el NAO reconoce a la persona, pero esta no es el usuario receptor, se realiza una pequeña rotación del NAO hacia la izquierda y se devuelve en el parámetro *recognise* el valor *false*. En cambio, si se trata del usuario receptor del mensaje, se devuelve en el parámetro *recognise* el valor *true*.

Los módulos utilizados en esta acción son: *ALTextToSpeech* (para comunicar frases a la persona encontrada), *ALFaceDetection* (para el reconocimiento de rostro de la persona), *ALMemory* (para acceder a la clave de memoria *FaceDetected* y borrar los datos de la clave de memoria *PeoplePerception/PeopleDetected*) y *ALMotion* (para realizar la rotación del NAO).

La acción ***playMessageLowLevel*** es la acción de bajo nivel encargada de reproducir el mensaje al usuario receptor y conocer si este quiere responder al mensaje recibido. En primer lugar, saluda al usuario receptor y le reproduce el mensaje grabado. Después, le pregunta al usuario si quiere responder al mensaje recibido. Si contesta que sí, se devuelve en el parámetro *answer* el valor *true* y si contesta que no, se coloca al NAO en la posición de sentado y se devuelve en el parámetro *answer* el valor *false*. Por último, se borran los valores de memoria necesarios.

Los módulos utilizados en esta acción son: *ALTextToSpeech* (para comunicar frases al usuario receptor), *ALAudioPlayer* (para reproducir el mensaje grabado), *ALSpeechRecognition* (para conocer si se quiere responder al mensaje), *ALMemory* (para acceder a la clave de memoria *WordRecognized* y borrar los datos de la clave de memoria *PeoplePerception/PeopleDetected*), *ALRobotPosture* (para colocar al NAO en la posición de sentado) y *ALTracker* (para detener el rastreo de marcas y borrar todos los objetivos a rastrear).

La acción ***rebuildRouteLowLevel*** es la acción de bajo nivel encargada de hacer comunicar al NAO que se está reconstruyendo la ruta para devolver el mensaje. Comunica que la ruta para devolver el mensaje se está reconstruyendo y devuelve en el parámetro *rebuild* el valor *true*.

El único módulo utilizado en esta acción es *ALTextToSpeech* (para comunicar que se está reconstruyendo la ruta para devolver el mensaje).

Para la parte de comunicación con PELEA, se empleó la librería *socket* (que se comunica con el socket utilizado en PELEA), utilizando el método *recv* para recoger la petición del cliente (PELEA) y el método *send* para mandar la respuesta al cliente (PELEA).

5.6 Alternativas de solución

Durante el desarrollo del trabajo, surgieron momentos en los que hubo que decantarse entre diferentes alternativas de solución. Estas alternativas son las siguientes:

- **Elección del robot:** durante la reunión con el tutor donde se especificaron los objetivos a alcanzar con el trabajo, hubo dudas sobre qué robot utilizar para el trabajo. Las dos alternativas de robots que se manejaban eran el Pioneer y el NAO. Para el tema de la navegación en un entorno, seguramente el Pioneer habría sido una mejor alternativa, ya que posee ruedas para su movimiento, lo que lo facilita enormemente. Además, aunque puede hacerlo, la movilidad no es un punto fuerte del NAO, ya que en ocasiones presenta cierta falta de equilibrio, aparte de que su velocidad de avance es bastante reducida. Sin embargo, lo que terminó favoreciendo al NAO para su elección en este trabajo era la mayor variedad de funcionalidades que ofrecía con respecto al Pioneer, ya que el único aspecto en el que este mejoraba al NAO era la navegación en un entorno.
- **Elección del lenguaje del servidor:** aunque con PELEA se tuvo que trabajar con Java porque era el lenguaje en el que esta arquitectura estaba implementada, lo cierto es que se valoraron diferentes alternativas para la elección del lenguaje del servidor que recibía las peticiones de PELEA. En concreto, se tuvieron dos lenguajes para decidir: C++ y Python. En dicha elección, basándose en la experiencia anterior del alumno con estos lenguajes y en los consejos de la documentación de *Aldebaran Robotics* [42], se decidió utilizar Python, que además tiene una notación bastante más sencilla que la utilizada por C++.
- **Protección del fichero de usuarios autorizados:** en este caso hubo dos alternativas para utilizar el fichero *Users.cfg* en el sistema. La primera alternativa era utilizar este fichero sin ninguna protección y la segunda era cifrar su contenido. Valorando ambas alternativas, se llegó a la conclusión de que carecía de sentido que existiera un fichero de usuarios autorizados sin cifrar, ya que un usuario no autorizado podría leerlo para hacerse pasar por quién quisiera o modificarlo para añadirse a sí mismo como un usuario autorizado. Por este motivo, la primera alternativa fue descartada rápidamente y se optó por la segunda, donde el usuario, al estar el fichero cifrado, no podrá leer ni modificar dicho fichero para poder utilizar el sistema. Además, tampoco podrá descifrar el fichero ya que para eso necesita saber la contraseña con la que este fue cifrado.
- **Modo de acceso al sistema:** en el momento en el que el sistema desarrollado inicia su ejecución, se valoraron también dos alternativas diferentes para identificar al usuario que utiliza el sistema. La primera alternativa era pedir al usuario que dijera su nombre al NAO y a partir de aquí el sistema continuara su ejecución normal. La segunda

alternativa era pedir al usuario en vez de su nombre, su rostro. En el momento de decantarse por una de estas dos alternativas, se eligió la que se consideró que era más segura, por lo que se eligió la segunda alternativa. Utilizando la primera alternativa, si un usuario normal se entera del nombre del usuario administrador y se lo dice al NAO cuando inicia el sistema, estará entrando como un usuario administrador, lo que supone una grave brecha en la seguridad del sistema, ya que puede realizar cualquiera de las funciones que los usuarios normales tienen restringidas como, por ejemplo, borrar todos los rostros de la base de datos del NAO. Sin embargo, con la segunda alternativa, es mucho más complicado hacerse pasar por otra persona. La única opción que se podría intentar explotar en este caso es utilizar una foto de la persona en cuestión. Pero debido a que dicha persona guardó su rostro utilizando su cara y no una foto, la foto tendría que parecerse demasiado en tamaño y forma a la cara que puso la persona cuando guardó su rostro para que el usuario impostor pudiera acceder al sistema.

6. PRUEBAS DEL SISTEMA

El objetivo de este capítulo es mostrar el entorno donde se han realizado las pruebas del sistema y especificar un plan de pruebas que demuestra que todas las necesidades que se han planteado han sido resueltas de manera satisfactoria.

6.1 Entorno de pruebas

Todas las pruebas realizadas en este capítulo se han llevado a cabo en el laboratorio 2.1.B16 del Edificio Sabatini perteneciente a la Universidad Carlos III de Madrid, en el campus de Leganés.

Como ventajas, este laboratorio ofrece todos los materiales necesarios para poder trabajar con el NAO (routers, cables Ethernet…), además de tener bastante luminosidad, pero presenta un suelo bastante irregular que ha provocado algunas caídas del NAO durante el periodo de pruebas.

6.2 Pruebas realizadas

Para comprobar que el sistema implementado realiza bien todas las funciones requeridas, se han realizado una serie de pruebas. Para describir cada una de las pruebas, se ha seguido el siguiente modelo de tabla:

PR-XXX	
Nombre	
Descripción	
Resultado esperado	
Resultado obtenido	

Tabla 64: Formato de prueba

En la tabla anterior, podemos observar los siguientes campos:

- **Identificador:** código único que identifica a la prueba y que se compone de las siguientes partes:
  - **PR:** indica que se trata de una prueba.
  - **XXX:** indica el número único (identificador) que tiene la prueba.
- **Nombre:** título que da una breve información acerca del objetivo de la prueba.
- **Descripción:** explicación detallada de la prueba.

- **Resultado esperado:** resultado que se espera obtener con la realización de la prueba.
- **Resultado obtenido:** resultado que se obtiene al realizar la prueba.

Cada una de las pruebas realizadas son las siguientes:

PR-001	
Nombre	Acceso al sistema sin usuarios autorizados
Descripción	Un usuario trata de acceder al sistema sin que haya usuarios guardados en el fichero de usuarios autorizados.
Resultado esperado	El sistema termina su ejecución.
Resultado obtenido	Correcto.

Tabla 65: PR-001, Acceso al sistema sin usuarios autorizados

PR-002	
Nombre	Acceso al sistema de usuario no registrado
Descripción	Un usuario sin su rostro guardado trata de acceder al sistema.
Resultado esperado	El sistema comprueba el rostro del usuario, no le permite su acceso y termina su ejecución.
Resultado obtenido	Correcto.

Tabla 66: PR-002, Acceso al sistema de usuario sin rostro

PR-003	
Nombre	Acceso al sistema de usuario no autorizado con rostro
Descripción	Un usuario no autorizado, pero con su rostro guardado en el NAO, trata de acceder al sistema.
Resultado esperado	El sistema comprueba el rostro del usuario, verifica si está en el fichero de usuarios autorizados, no le permite su acceso y termina su ejecución.
Resultado obtenido	Correcto.

Tabla 67: PR-003, Acceso al sistema de usuario no autorizado con rostro

PR-004	
Nombre	Acceso al sistema de usuario autorizado con rostro, pero con formato de correo incorrecto
Descripción	Un usuario autorizado con su rostro guardado, pero con el formato

	de su correo siendo incorrecto, trata de acceder al sistema.
Resultado esperado	El sistema comprueba el rostro del usuario, verifica si está correctamente en el fichero de usuarios autorizados, no le permite su acceso y termina su ejecución.
Resultado obtenido	Correcto.

Tabla 68: PR-004, Acceso al sistema de usuario autorizado con rostro, pero con formato de correo incorrecto

PR-005	
Nombre	Acceso al sistema de usuario autorizado administrador con rostro registrado
Descripción	Un usuario autorizado administrador con su rostro guardado trata de acceder al sistema.
Resultado esperado	El sistema comprueba el rostro del usuario, verifica si está correctamente en el fichero de usuarios autorizados, le permite el acceso y le posibilita realizar cualquiera de las funciones que tiene el sistema.
Resultado obtenido	Correcto.

Tabla 69: PR-005, Acceso al sistema de usuario autorizado administrador con rostro

PR-006	
Nombre	Acceso al sistema de usuario autorizado normal con rostro registrado
Descripción	Un usuario autorizado normal con su rostro guardado trata de acceder al sistema.
Resultado esperado	El sistema comprueba el rostro del usuario, verifica si está correctamente en el fichero de usuarios autorizados, le permite el acceso y le posibilita realizar las funciones de comunicarse y salir.
Resultado obtenido	Correcto.

Tabla 70: PR-006, Acceso al sistema de usuario autorizado normal con rostro

PR-007	
Nombre	Función del sistema incorrecta
Descripción	El usuario no responde correctamente a la pregunta sobre la función que quiere realizar.
Resultado esperado	El sistema preguntará al usuario sobre la función que quiere realizar hasta que responda correctamente.

Resultado obtenido	Correcto.

Tabla 71: PR-007, Función del sistema incorrecta

PR-008	
Nombre	Aprendizaje de rostro de usuario no autorizado
Descripción	El administrador trata de guardar el rostro de un usuario que no está autorizado en la base de datos del NAO.
Resultado esperado	El sistema no permite guardar el rostro y termina su ejecución.
Resultado obtenido	Correcto.

Tabla 72: PR-008, Aprendizaje de rostro de usuario no autorizado

PR-009	
Nombre	Aprendizaje de rostro repetido
Descripción	El administrador trata de guardar el rostro de un usuario que ya tiene su rostro guardado en la base de datos del NAO.
Resultado esperado	El sistema no permite guardar el rostro y termina su ejecución.
Resultado obtenido	Correcto.

Tabla 73: PR-009, Aprendizaje de rostro repetido

PR-010	
Nombre	Aprendizaje de rostro
Descripción	El administrador quiere guardar el rostro de un usuario en la base de datos del NAO.
Resultado esperado	El sistema guarda el rostro y termina su ejecución.
Resultado obtenido	Correcto.

Tabla 74: PR-010, Aprendizaje de rostro

PR-011	
Nombre	Búsqueda de rostro no realizable
Descripción	El administrador trata de buscar un rostro sin que haya rostros guardados en la base de datos del NAO.

Resultado esperado	El sistema no permite la búsqueda del rostro y termina su ejecución.
Resultado obtenido	Correcto.

Tabla 75: PR-011, Búsqueda de rostro no realizable

PR-012	
Nombre	Búsqueda de rostro
Descripción	El administrador quiere buscar si un rostro está almacenado en la base de datos del NAO o no.
Resultado esperado	En caso de que el rostro esté guardado el sistema informará de que el rostro existe en la base de datos del NAO y si no está guardado viceversa, terminando su ejecución en ambos casos.
Resultado obtenido	Correcto.

Tabla 76: PR-012, Búsqueda de rostro

PR-013	
Nombre	Listado de todos los rostros no realizable
Descripción	El administrador trata de listar todos los rostros sin que haya rostros guardados en la base de datos del NAO.
Resultado esperado	El sistema no permite el listado de todos los rostros y termina su ejecución.
Resultado obtenido	Correcto.

Tabla 77: PR-013, Listado de todos los rostros no realizable

PR-014	
Nombre	Listado de todos los rostros
Descripción	El administrador quiere listar todos los rostros guardados en la base de datos del NAO.
Resultado esperado	El sistema lista todos los rostros guardados y termina su ejecución.
Resultado obtenido	Correcto.

Tabla 78: PR-014, Listado de todos los rostros

PR-015	
Nombre	Eliminación de rostro no realizable
Descripción	El administrador trata de eliminar un rostro sin que haya rostros guardados en la base de datos del NAO.
Resultado esperado	El sistema no permite la eliminación del rostro y termina su ejecución.
Resultado obtenido	Correcto.

Tabla 79: PR-015, Eliminación de rostro no realizable

PR-016	
Nombre	Eliminación de rostro inexistente
Descripción	El administrador trata de eliminar un rostro que no existe en la base de datos del NAO.
Resultado esperado	El sistema no permite la eliminación del rostro y termina su ejecución.
Resultado obtenido	Correcto.

Tabla 80: PR-016, Eliminación de rostro inexistente

PR-017	
Nombre	Eliminación de rostro
Descripción	El administrador quiere eliminar un rostro de la base de datos del NAO.
Resultado esperado	El sistema elimina el rostro y termina su ejecución.
Resultado obtenido	Correcto.

Tabla 81: PR-017, Eliminación de rostro

PR-018	
Nombre	Eliminación de todos los rostros no realizable
Descripción	El administrador trata de eliminar todos los rostros sin que haya rostros guardados en la base de datos del NAO.
Resultado esperado	El sistema no permite la eliminación de todos los rostros y termina su ejecución.
Resultado obtenido	Correcto.

Tabla 82: PR-018, Eliminación de todos los rostros no realizable

PR-019	
Nombre	Eliminación de todos los rostros
Descripción	El administrador quiere eliminar todos los rostros de la base de datos del NAO.
Resultado esperado	El sistema elimina todos los rostros y termina su ejecución.
Resultado obtenido	Correcto.

Tabla 83: PR-019, Eliminación de todos los rostros

PR-020	
Nombre	Usuario receptor no autorizado
Descripción	El usuario receptor del que se dice el nombre es un usuario no autorizado.
Resultado esperado	El sistema no permite mandar un mensaje a dicho usuario receptor y termina la ejecución.
Resultado obtenido	Correcto.

Tabla 84: PR-020, Usuario receptor no autorizado

PR-021	
Nombre	Usuario receptor con formato de correo incorrecto
Descripción	El correo del usuario receptor del que se dice el nombre tiene un formato incorrecto en el fichero de usuarios autorizados.
Resultado esperado	El sistema no permite mandar un mensaje a dicho usuario receptor y termina la ejecución.
Resultado obtenido	Correcto.

Tabla 85: PR-021, Usuario receptor con formato de correo incorrecto

PR-022	
Nombre	Duración del mensaje incorrecta
Descripción	El usuario emisor elige una duración del mensaje que no es ni de diez, ni de treinta ni de sesenta segundos.
Resultado esperado	El sistema preguntará al usuario emisor una duración hasta que este elija una de las tres opciones.
Resultado obtenido	Correcto.

Tabla 86: PR-022, Duración del mensaje incorrecta

PR-023	
Nombre	Tipo de mensaje incorrecto
Descripción	El usuario emisor no responde correctamente a la pregunta de si el mensaje es personal o no.
Resultado esperado	El sistema preguntará al usuario emisor sobre el tipo de mensaje hasta que responda correctamente.
Resultado obtenido	Correcto.

Tabla 87: PR-023, Tipo de mensaje incorrecto

PR-024	
Nombre	Usuario receptor sin rostro
Descripción	El usuario emisor elige enviar el mensaje haciendo que lo entregue el propio NAO, pero el usuario receptor no tiene su rostro almacenado.
Resultado esperado	El sistema no permite entregar un mensaje a dicho usuario receptor y termina la ejecución.
Resultado obtenido	Correcto.

Tabla 88: PR-024, Usuario receptor sin rostro

PR-025	
Nombre	Pérdida de marca
Descripción	El NAO pierde de vista una marca que había detectado.
Resultado esperado	El sistema hará que el NAO se disponga a buscar de nuevo la marca perdida rotando sobre sí mismo.
Resultado obtenido	Correcto.

Tabla 89: PR-025, Pérdida de marca

PR-026	
Nombre	Caída del NAO
Descripción	El NAO se cae al suelo durante la navegación y pierde de vista una marca que había detectado.
Resultado esperado	El sistema hará que el NAO se levante y se disponga a buscar de nuevo la marca perdida rotando sobre sí mismo.
Resultado obtenido	Correcto.

Tabla 90: PR-026, Caída del NAO

PR-027	
Nombre	Ausencia de persona
Descripción	El NAO no ve a ninguna persona.
Resultado esperado	El sistema hará que el NAO rote sobre sí mismo y compruebe si ve a alguna persona.
Resultado obtenido	Correcto.

Tabla 91: PR-027, Ausencia de persona

PR-028	
Nombre	Usuario receptor incorrecto
Descripción	El NAO encuentra a una persona, pero esta no es el usuario receptor.
Resultado esperado	El sistema hará que el NAO rote sobre sí mismo y compruebe si ve a alguna persona.
Resultado obtenido	Correcto.

Tabla 92: PR-028, Usuario receptor incorrecto

PR-029	
Nombre	Respuesta al mensaje incorrecta
Descripción	El usuario receptor no responde correctamente a la pregunta de si quiere responder al mensaje que acaba de recibir.
Resultado esperado	El sistema preguntará al usuario receptor si quiere responder al mensaje hasta que responda correctamente.
Resultado obtenido	Correcto.

Tabla 93: PR-029, Respuesta al mensaje incorrecta

PR-030	
Nombre	Entrega de mensaje por correo al usuario receptor
Descripción	Se comprueba que el sistema es capaz de mandar por correo un mensaje al usuario receptor.
Resultado esperado	El sistema entrega, mediante un correo, un mensaje al usuario receptor.
Resultado obtenido	Correcto.

Tabla 94: PR-030, Entrega de mensaje por correo al usuario receptor

PR-031	
Nombre	Entrega de mensaje por el propio NAO al usuario receptor
Descripción	Se comprueba que el sistema es capaz de hacer entregar con el propio NAO un mensaje al usuario receptor.
Resultado esperado	El sistema dice, utilizando al NAO, el mensaje al usuario receptor.
Resultado obtenido	Correcto.

Tabla 95: PR-031, Entrega de mensaje por el propio NAO al usuario receptor

PR-032	
Nombre	Entrega de mensaje por correo al usuario emisor
Descripción	Se comprueba que tras entregar el propio NAO el mensaje al usuario receptor, el sistema es capaz de mandar por correo la respuesta al usuario emisor original.
Resultado esperado	El sistema dice, utilizando al NAO, el mensaje al usuario receptor y entrega, mediante un correo, la respuesta al usuario emisor.
Resultado obtenido	Correcto.

Tabla 96: PR-032, Entrega de mensaje por correo al usuario emisor

PR-033	
Nombre	Entrega de mensaje por el propio NAO al usuario emisor
Descripción	Se comprueba que tras entregar el propio NAO el mensaje al usuario receptor, el sistema es capaz de hacer que el NAO entregue de nuevo personalmente la respuesta al usuario emisor original.
Resultado esperado	El sistema dice, utilizando al NAO, el mensaje al usuario receptor y la respuesta al usuario emisor.
Resultado obtenido	Correcto.

Tabla 97: PR-033, Entrega de mensaje por el propio NAO al usuario emisor

PR-034	
Nombre	Salida del sistema
Descripción	El usuario tras iniciar el sistema quiere salir de él.
Resultado esperado	El sistema termina su ejecución.
Resultado obtenido	Correcto.

Tabla 98: PR-034, Salida del sistema

7. MARCO REGULADOR

El objetivo de este capítulo consiste en analizar las restricciones legales que pueden afectar al proyecto, enumerar las licencias de las herramientas utilizadas, describir los estándares técnicos utilizados y considerar aquellas cuestiones relacionadas con la propiedad intelectual del trabajo.

7.1 Legislación aplicable

El ámbito de la robótica siempre ha sido un campo difícil de valoración en el aspecto legal. Siempre existe el debate de si a un robot hay que aplicarle las mismas normas que a una persona o, por el contrario, deberían tener una normativa más específica.

En este tema, una película que enfoca bastante bien esta problemática es *Yo, robot* (2004). En ella, ambientada en el año 2035, los robots han alcanzado un gran avance hasta el punto de que están muy cerca de parecerse a los seres humanos.

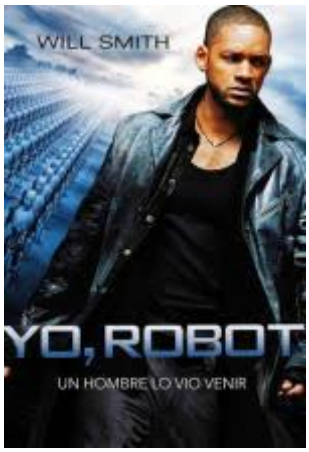


Ilustración 73: Película Yo robot

Con la idea de controlar a estos robots tan avanzados, todos ellos deben regirse por tres leyes desarrolladas para la robótica. Estas leyes son las siguientes:

- **Primera ley:** un robot no hará daño a un ser humano ni permitirá que un ser humano sufra daño alguno.
- **Segunda ley:** un robot debe cumplir las órdenes de un ser humano, siempre que estas órdenes no entren en conflicto con la primera ley.
- **Tercera ley:** un robot debe proteger su propia existencia, siempre que esta protección no entre en conflicto con la primera o la segunda ley.

Estas leyes realmente son las que creó el escritor de ciencia ficción *Isaac Asimov* [43], con las que se pretende regular el comportamiento de los robots de sus novelas y cuentos.

Ya hablando en el mundo real y actual, actualmente no existen unas normas parecidas que estén en aplicación, aunque hay que tener en cuenta que el nivel en el que se encuentra hoy en día la robótica no es ni mucho menos el alcanzado en la película mencionada.

Pero teniendo en cuenta el desarrollo que están teniendo tanto el campo de la robótica como el de la Inteligencia Artificial, la Unión Europea y algunos países como Estados Unidos han empezado a realizar borradores de lo que serían unas futuras leyes de la robótica.

Atendiendo a la Unión Europea, el 16 de febrero de 2017 el Parlamento Europeo aprobó una resolución sobre las leyes de la robótica en la Unión Europea [44], en el que se insta a la Unión Europea a desarrollar una legislación específica sobre el uso de la Inteligencia Artificial y sus límites.

Atendiendo a esta resolución, existen una serie de puntos clave que deberán guiar la normativa.

En primer lugar, se deben de tener en cuenta las características que determinan si un robot es inteligente o no. Según la Unión Europea, un robot será inteligente cuando tenga alguna de las siguientes habilidades:

- Adquiere autonomía mediante sensores o intercambio de datos con su entorno.
- Tiene capacidad de autoaprendizaje.
- Tiene soporte físico.
- Adapta su comportamiento y sus acciones al entorno en el que se encuentra.

En segundo lugar, los investigadores en robótica deben respetar ciertos principios éticos durante la realización de su labor:

- **Principio de beneficencia:** los robots deben actuar en beneficio del hombre.
- **Principio de no maleficencia:** los robots no deben perjudicar a las personas.
- **Principio de autonomía:** los robots deben tener la capacidad de tomar decisiones con conocimiento de causa independientemente de la interacción con otros robots.
- **Principio de justicia:** distribución justa de los beneficios asociados a la robótica y asequibilidad de los robots utilizados en el ámbito de la asistencia sanitaria a domicilio y de los cuidados sanitarios en particular.

En definitiva, esta resolución se traduce en la aplicación de una serie de normas que permitan una correcta regulación de los robots [45]:

- **Interruptor de emergencia:** todos los robots desarrollados deberán tener un interruptor de emergencia que permita su apagado en una situación de peligro.
- **Protección del ser humano:** los robots no podrán realizar ningún daño a las personas.
- **Evitar relaciones emocionales con los robots:** al igual que los robots no son capaces de sentir amor por las personas, los seres humanos deben actuar del mismo modo con los robots.
- **Seguro obligatorio:** los dueños de los robots de mayor tamaño y, por lo tanto, con mayor probabilidad de provocar daños en el entorno, deberán contratar un seguro obligatorio con el objetivo de cubrir los posibles daños provocados por dichos robots.
- **Derechos y obligaciones:** los robots serán considerados como "personas electrónicas" y, por ello, tendrán una serie de derechos y deberán asumir las consecuencias de sus actos junto a sus dueños.
- **Pago de impuestos:** con el objetivo de subvencionar ayudas y reducir el impacto negativo de los despidos de personas en su trabajo por sustitución de un robot, los robots deberán tributar en la seguridad social.

Por otro lado, en el sistema que se ha desarrollado, se realiza el tratamiento y uso de datos personales, ya sea tanto el administrador al configurar y utilizar el fichero de usuarios autorizados para utilizar el sistema como el propio NAO al almacenar dentro de él el nombre de personas y sus rostros (de forma vectorizada esto último).

Considerando estas cuestiones, es necesario tener en cuenta en este trabajo la *Ley Orgánica 15/1999, de 13 de diciembre, de Protección de Datos de Carácter Personal* (LOPD).

Pero debido a que el 25 de mayo de 2018 fue de aplicación el *Reglamento General de Protección de Datos*, una vez que entró en vigor el 25 de mayo de 2016, la legislación española sustituyó la ley de protección de datos mencionada anteriormente por la *Ley Orgánica 3/2018, de 5 de diciembre, de Protección de Datos Personales y garantía de los derechos digitales* (LOPD-GDD) [46], que es la que se tendrá en consideración.

Por tanto, en primer lugar, con la idea de proteger la confidencialidad de los datos almacenados en este sistema se tomarán medidas tanto con el robot NAO como con el fichero de usuarios, que son las fuentes de los datos personales.

Para la protección de los datos almacenados dentro del NAO, este robot mientras no esté en uso se encontrará guardado y protegido con el objetivo de que personas no autorizadas no puedan acceder a la información que contiene.

Para la protección de los datos almacenados en el fichero de usuarios autorizados para utilizar el sistema, se ha optado por su cifrado con la herramienta GnuPG, con el objetivo de que ningún usuario que no sea el administrador pueda leer ni modificar los datos almacenados en el fichero. Así pues, la contraseña que se utilizó en el cifrado del fichero deberá ser conocida únicamente por el usuario administrador del sistema.

En segundo lugar, con respecto a este usuario administrador, al ser el responsable del tratamiento de los datos personales, permitirá a los usuarios que utilizan el sistema los siguientes derechos de la LOPD-GDD que aplican en este caso:

- **Derecho de acceso:** el administrador permitirá a un usuario autorizado ver los datos personales propios que están siendo utilizados en el sistema.
- **Derecho de oposición y supresión:** el usuario autorizado puede oponerse a que sus datos personales sean utilizados en el sistema, debiendo el administrador responsable realizar la eliminación de estos.
- **Derecho a la limitación del tratamiento:** el usuario autorizado en este caso tiene derecho a pedir al administrador que quite alguno de sus datos personales que están siendo utilizados en el sistema, pero dado que todos estos datos son necesarios para la utilización del sistema por parte del usuario, este último estaría renunciando a la utilización de dicho sistema.

En tercer y último lugar, el usuario administrador del sistema deberá:

- Informar a los usuarios de los que trate datos personales en el sistema de su utilización, con el objetivo de que estos usuarios sean conscientes del uso de sus datos personales en el sistema de transmisión de mensajes.
- Utilizar los datos personales que tiene el sistema únicamente el ámbito del sistema desarrollado, no pudiendo comunicar ninguno de estos datos personales a nadie distinto al propio usuario propietario de los datos.

7.2 Licencias de las herramientas empleadas

Durante el desarrollo del trabajo se han utilizado herramientas o lenguajes de programación que en algunos casos están sujetos a licencias:

- **Productos de Microsoft:** esto incluye el uso tanto de Word (Office) para la redacción de esta memoria como de Project para la planificación del proyecto. La versión de Office utilizada es de ámbito educativo y se le ha proporcionado al alumno a través de cuenta en la universidad. La versión de Project también es de ámbito educativo, pero no se ha llegado a instalar en el ordenador, sino que se han utilizado ordenadores de la universidad para su uso.
- **Java:** se encuentra bajo la licencia *GNU General Public License*, que es una licencia de software de código abierto.
- **Eclipse:** se encuentra bajo la licencia *Eclipse Public License*, que es también una licencia de software de código abierto.
- **Python:** se encuentra bajo la licencia *Python Software Foundation License*, que es una licencia de software libre permisiva, es decir, su distribución no está sujeta a ninguna licencia, permitiendo modificaciones del código fuente y la creación de trabajos derivados sin necesidad de que estos tengan que ser a su vez código libre también.
- **Spyder:** se encuentra bajo una licencia MIT (*Massachusetts Institute of Technology*), que es también una licencia de software libre permisiva.
- **GnuPG:** se encuentra, al igual que Java, bajo la licencia *GNU General Public License*.
- **Pelea:** se encuentra bajo una licencia *Creative Commons*, por lo que tiene un carácter gratuito.
- **Choregraphe:** se encuentra bajo una licencia EULA (*End User License Agreement*), que es una licencia donde *Aldebaran Robotics* especifica los artículos que el usuario final debe respetar durante la utilización del software.

7.3 Estándares técnicos

En cuanto a los estándares técnicos utilizados en el trabajo, únicamente se ha utilizado el siguiente:

- **Formato IEEE:** se ha utilizado para la realización de las referencias de este trabajo. Se ha empleado dicho formato porque es el recomendado en los proyectos de ingeniería.

7.4 Propiedad intelectual

El software que ha sido desarrollado, como se detallará en el siguiente capítulo, está orientado a la investigación, por lo que el objetivo es que tenga una licencia *Creative Commons*, permitiendo así a cualquiera de los estudiantes o personal docente de la Universidad Carlos III de Madrid utilizarlo de manera gratuita.

8. ENTORNO SOCIO-ECONÓMICO

El objetivo de este capítulo es mostrar la planificación que se ha realizado del proyecto, el presupuesto asociado al trabajo y el impacto socio-económico que se espera de la realización del proyecto.

8.1 Planificación

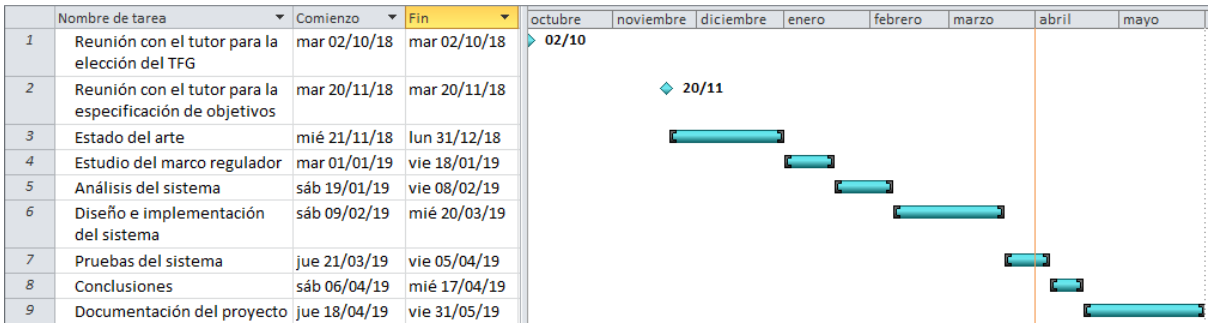
En este apartado se muestra la planificación que se ha realizado del proyecto, explicando cada una de las fases de las que se compone y ubicándolas temporalmente mediante la realización de un diagrama de Gantt tanto para la planificación estimada como para la planificación real.

La realización del proyecto se compone de las siguientes fases:

- **Reunión con el tutor para la elección del TFG:** en esta fase se realiza una reunión del alumno con el tutor para que este le presente los distintos trabajos que tiene disponibles.
- **Reunión con el tutor para la especificación de objetivos:** en esta fase se realiza una reunión del alumno con el tutor para que se especifiquen los objetivos que se deberán alcanzar con el trabajo.
- **Estado del arte:** en esta fase se realiza una búsqueda de información acerca de la robótica, la planificación automática y la arquitectura PELEA. Además, se busca información sobre aplicaciones actuales de robots en la navegación en un entorno y en el reconocimiento facial.
- **Estudio del marco regulador:** en esta fase se realiza un estudio de las restricciones legales que podría tener el proyecto.
- **Análisis del sistema:** en esta fase se definen tanto los casos de uso como los requisitos que el sistema deberá cumplir.
- **Diseño e implementación del sistema:** en esta fase se realiza el diseño con el que se define la arquitectura que tendrá el sistema y la implementación que cumple dicho diseño.
- **Pruebas del sistema:** en esta fase se desarrollan las pruebas que demuestran el correcto funcionamiento del sistema.
- **Conclusiones:** en esta fase se razonan las conclusiones obtenidas con la realización del trabajo.

- **Documentación del proyecto:** última fase del proyecto, donde se finalizan aquellos apartados de la documentación del proyecto que queden pendientes.

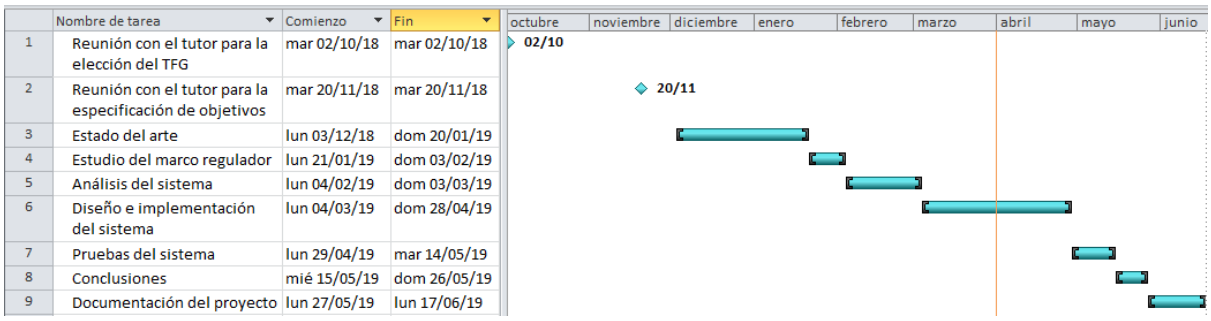
Una vez que las fases del proyecto ya han sido descritas, se utiliza un diagrama de Gantt con el objetivo de mostrar la ubicación temporal estimada de cada una de ellas, disponiendo entonces de una planificación estimada del proyecto:



	Nombre de tarea	Comienzo	Fin
1	Reunión con el tutor para la elección del TFG	mar 02/10/18	mar 02/10/18
2	Reunión con el tutor para la especificación de objetivos	mar 20/11/18	mar 20/11/18
3	Estado del arte	mié 21/11/18	lun 31/12/18
4	Estudio del marco regulador	mar 01/01/19	vie 18/01/19
5	Análisis del sistema	sáb 19/01/19	vie 08/02/19
6	Diseño e implementación del sistema	sáb 09/02/19	mié 20/03/19
7	Pruebas del sistema	jue 21/03/19	vie 05/04/19
8	Conclusiones	sáb 06/04/19	mié 17/04/19
9	Documentación del proyecto	jue 18/04/19	vie 31/05/19

Ilustración 74: Planificación inicial del proyecto

Con el desarrollo del proyecto, la planificación anterior sufrió cambios debido a distintos motivos: días que no se pudieron dedicar al Trabajo de Fin de Grado, mayor duración de una tarea que la estimada o, por el contrario, que una tarea al final ha llevado menos tiempo del que se estimó. Teniendo en cuenta estas modificaciones, el diagrama de Gantt con la planificación real queda de la siguiente forma:



	Nombre de tarea	Comienzo	Fin
1	Reunión con el tutor para la elección del TFG	mar 02/10/18	mar 02/10/18
2	Reunión con el tutor para la especificación de objetivos	mar 20/11/18	mar 20/11/18
3	Estado del arte	lun 03/12/18	dom 20/01/19
4	Estudio del marco regulador	lun 21/01/19	dom 03/02/19
5	Análisis del sistema	lun 04/02/19	dom 03/03/19
6	Diseño e implementación del sistema	lun 04/03/19	dom 28/04/19
7	Pruebas del sistema	lun 29/04/19	mar 14/05/19
8	Conclusiones	mié 15/05/19	dom 26/05/19
9	Documentación del proyecto	lun 27/05/19	lun 17/06/19

Ilustración 75: Planificación final del proyecto

Aquellas fases que sufrieron modificaciones desde la planificación estimada hasta la planificación real son las siguientes:

- **Estado del arte:** esta fase sufrió retrasos en su estimación debido a que la búsqueda de las fuentes que se utilizan en este apartado llevó más tiempo del esperado.
- **Estudio del marco regulador:** esta fase llevó menos tiempo del planificado debido a que las restricciones legales que aplicaban a este proyecto no eran tantas como el alumno esperaba al inicio del proyecto.

- **Análisis del sistema:** esta fase llevó más tiempo del planificado debido a que se tuvo que realizar varias versiones de los casos de uso y de los requisitos hasta que se obtuvo una especificación correcta de ellos.
- **Diseño e implementación del sistema:** esta fase es sin duda la que más retraso ha sufrido con respecto a lo planificado al inicio del proyecto. Esto se ha producido debido a que el alumno tuvo que aprender desde cero cómo trabajar con el NAO y con PELEA, para después trabajar con ambos conjuntamente.
- **Documentación del proyecto:** debido al retraso acumulado de las fases anteriores, el inicio de esta fase se tuvo que retrasar y, debido a la fecha de entrega de la memoria del proyecto, se dedicaron muchos menos días de los planificados para completar esta fase.

En la siguiente tabla se muestran los días y las horas de cada una de las fases que componen el proyecto, tanto para la planificación estimada como para la planificación real.

Fase	Días estimados	Horas estimadas	Días reales	Horas reales
Reunión con el tutor para la elección del TFG	1	1	1	1
Reunión con el tutor para la especificación de objetivos	1	1	1	1
Estado del arte	41	164	49	196
Estudio del marco regulador	18	72	14	42
Análisis del sistema	21	63	28	112
Diseño e implementación del sistema	40	160	56	280
Pruebas del sistema	16	48	16	48
Conclusiones	12	11	12	11
Documentación del proyecto	44	66	22	77
Total	**194**	**586**	**199**	**768**

Tabla 99: Registro de horas del proyecto

8.2 Presupuesto

En este apartado se recoge el análisis de los gastos y costes internos que supondrá el desarrollo del proyecto.

En primer lugar, se detalla el coste del personal que ha trabajado en el proyecto. Dado que únicamente ha trabajado en el proyecto el alumno que realiza este Trabajo de Fin de Grado, sólo se tendrá en cuenta a este en el cálculo del coste del personal.

Las horas dedicadas al proyecto han sido obtenidas en el apartado anterior del número total de horas reales dedicadas al proyecto.

El coste por hora ha sido obtenido del informe sobre el estado del mercado laboral en España elaborado por *InfoJobs* en el año 2017 [47], en la sección en la que se habla del salario en el ámbito de la robótica, y del límite de horas anual estipulado por ley [48].

Teniendo en cuenta estas dos fuentes, el salario anual está en torno a los 29.784 € y el máximo de horas laborales por año es de aproximadamente 1.826 horas. Con estos dos factores, se obtiene un coste por hora de 16,31 €.

Nombre	Horas dedicadas al proyecto	Coste/hora
David Gutiérrez Fernández	768	16,31 €
Total		**12.526,08 €**

Tabla 100: Coste del personal

En segundo lugar, se detalla el coste material de los equipos utilizados en el proyecto, tanto en el aspecto hardware como en el aspecto software. Teniendo en cuenta que cada producto tiene su propia vida útil, el coste para el proyecto se obtendrá multiplicando el coste total del producto por la duración del proyecto, dividiendo este resultado entre la vida útil del producto.

Producto	Coste total	Vida útil (meses)	Periodo de uso (meses)	Coste para el proyecto
Ordenador portátil Toshiba Satellite C855	575,00 €	36	7	111,81 €
Router WiFi	15,98 €	48	7	2,33 €
Cable Ethernet	0,70 €	48	7	0,10 €
Robot NAO H25	7.999 €	60	7	933,22 €
SO Windows 10	145,00 €	36	7	28,19 €
Microsoft Office	89,99 €	72	7	8,75 €
Microsoft Project	118,50 €	72	7	11,52 €
SO Linux Ubuntu 18.04	0,00 €	36	7	0,00 €
Eclipse	0,00 €	60	7	0,00 €
JDK	0,00 €	60	7	0,00 €
PELEA	0,00 €	60	7	0,00 €
Choregraphe	0,00 €	60	7	0,00 €
Python	0,00 €	12	7	0,00 €
Spyder	0,00 €	60	7	0,00 €

GnuPG	0,00 €	60	7	0,00 €
Total				**1.095,92 €**

Tabla 101: Coste de los equipos

En tercer lugar, se detalla el coste de los materiales fungibles utilizados en el proyecto. El uso de estos va orientado a la toma de notas y a la impresión de las marcas que el NAO reconoce cuando está siguiendo una ruta.

Producto	Descripción	Unidades	Coste
Paquete de 100 folios	Se utilizará para imprimir las marcas que el NAO debe reconocer	1	1,99 €
Cuaderno de tamaño cuartilla de 80 hojas	Se utilizará para la toma de notas	1	1,00 €
Bolígrafo BIC	Se utilizará para la toma de notas	3	0,90 €
Total			**3,89 €**

Tabla 102: Coste de los materiales fungibles

En cuarto lugar, se detalla el coste de los desplazamientos y dietas realizados durante el proyecto. En lo relativo al desplazamiento, el alumno ha realizado sus desplazamientos en transporte público, por lo que únicamente ha necesitado tener activo un abono de transporte. En lo relativo a las dietas, no se ha realizado ninguna dieta durante el desarrollo del proyecto, por lo que este concepto no se tiene en cuenta.

Concepto	Coste/Mes	Periodo de uso (meses)	Coste para el proyecto
Abono transporte joven	20,00 €	7	140,00 €
Total			**140,00 €**

Tabla 103: Coste de los desplazamientos y dietas

En quinto lugar, se detallan los costes indirectos asociados a la realización del proyecto. Dichos costes han sido obtenidos de las facturas correspondientes. Sin embargo, para obtener un coste que se ajustara más a la realidad, teniendo en cuenta que el alumno vive con sus padres, se ha dividido cada uno de los importes de las facturas entre tres.

Concepto	Coste/Mes	Periodo de uso (meses)	Coste para el proyecto
Internet	23,00 €	7	161,00 €
Agua	9,04 €	7	63,28 €
Electricidad	20,47 €	7	143,29 €
Total			**367,57 €**

Tabla 104: Costes indirectos

En sexto y último lugar, se realiza la suma de todos los costes asociados al proyecto, aplicando a dicho resultado un margen de riesgo de un diez por ciento, un margen de beneficio de un veinte por ciento y el coste asociado al IVA (veintiuno por ciento).

El margen de riesgo se aplica sobre el coste total del proyecto, mientras que el margen de beneficio se aplica a la suma del coste total del proyecto y el margen de riesgo.

Concepto	Valor	Coste acumulado
Coste del personal	12.526,08 €	12.526,08 €
Coste de los equipos	1.095,92 €	13.622,00 €
Coste de los materiales fungibles	3,89 €	13.625,89 €
Coste de los desplazamientos y dietas	140,00 €	13.765,89 €
Costes indirectos	367,57 €	14.133,46 €
Coste total		**14.133,46 €**
Margen de riesgo (10%)	1.413,35 €	15.546,81 €
Margen de beneficio (20%)	3.109,36 €	18.656,17 €
IVA (21%)	3.917,80 €	22.573,97 €
Coste final		**22.573,97 €**

Tabla 105: Coste final del proyecto

Por tanto, el coste final que tendría el proyecto sería de **22.573,97 €**.

8.3 Impacto socio-económico

En este apartado se evalúa el impacto que tendrá el sistema desarrollado teniendo en cuenta los aspectos económico, social y ético. El impacto medioambiental ha sido omitido puesto que carece de importancia en este trabajo.

En el aspecto económico, se ha de tener en cuenta que el trabajo desarrollado no está orientado obtener ningún beneficio de él, sino que está orientado a fines de investigación.

En el aspecto social, se deben valorar los posibles usos que tendría el sistema desarrollado. Dado que es un sistema que permite la comunicación entre dos personas que se encuentran en distintas ubicaciones, un posible uso de este sistema sería su utilización por parte de personas que tienen problemas de movilidad y que, o bien no se pueden desplazar, o bien ese desplazamiento les requeriría mucho más esfuerzo que a una persona sana.

Por poner un ejemplo, una persona que se encuentre en una silla de ruedas podría mandar al NAO en su lugar para que transmitiera un mensaje que no es personal a otra persona que se encuentra a una cierta distancia física. Con ello, se tendría un impacto social positivo, ya que permitiría a personas que o bien no se pueden desplazar o bien el esfuerzo que deben hacer al desplazarse es demasiado grande comunicarse con otras personas que se encuentren en ubicaciones distintas.

En el aspecto ético, se ha de mencionar que todos los datos que se utilizarán con este sistema cumplirán con la normativa recogida en la *Ley Orgánica 3/2018, de 5 de diciembre, de Protección de Datos Personales y garantía de los derechos digitales* (LOPD-GDD).

9. CONCLUSIONES Y TRABAJOS FUTUROS

El objetivo de este capítulo es mostrar los objetivos que se han alcanzado en el Trabajo de Fin de Grado, las conclusiones obtenidas con su realización y las posibles futuras líneas de trabajo que ofrece.

9.1 Objetivos cumplidos

Cuando se explicaron los objetivos del sistema que se pretendía crear en el tercer capítulo de este trabajo, se tenía claro las metas a alcanzar, pero no del todo la forma de conseguir estas.

El sistema partía de dos posibles formas de entregar un mensaje. Con respecto a la entrega por correo, sin duda era la vía más sencilla de las dos planificadas. Su implementación fue bastante rápida comparada con la realizada en la segunda vía de entrega. La idea era utilizar un correo electrónico creado específicamente para este Trabajo de Fin de Grado, y a partir de él, adjuntar en un correo el audio con el mensaje grabado y mandarlo al correo electrónico del usuario receptor correspondiente.

Sin embargo, la entrega del mensaje por el propio NAO presentó bastantes dificultades. La parte de identificar a una persona, en este caso al usuario receptor del mensaje, se pudo cubrir bien con las funcionalidades que el NAO ofrecía. En cambio, la navegación en un entorno para localizar al usuario receptor fue una de las partes más complicadas del trabajo, ya que en la robótica es un tema complejo de abordar.

Se plantearon los distintos problemas que la navegación implicaba, como los posibles obstáculos presentes en la ruta que hiciera el robot o el cómo controlar la posición del robot en la ruta. Con esta situación, investigando, se encontraron las marcas que el NAO reconocía y que hacían posible una navegación guiada por pistas, siendo estas pistas las marcas reconocibles por el NAO. Con ello, la dificultad de ir controlando la posición del robot durante la navegación pudo ser solventada.

Con respecto al tema de los obstáculos, es seguramente una de las vías con más potencial para mejorar este trabajo y aumentar su complejidad, dado que con la implementación realizada el NAO requiere que durante la ruta a realizar no haya obstáculos que impidan realizar el recorrido.

Pero también es cierto, que en el trabajo se han tenido en cuenta situaciones más que posibles durante la navegación, como son el hecho de perder de vista una marca que se está intentando

alcanzar o el hecho de que el NAO pierda el equilibrio durante su movimiento y se caiga al suelo. Por ello, es posible que, aunque se colocaran obstáculos que pudieran provocar incluso la caída del NAO, este se levantara y acabara superando con éxito los obstáculos concretos.

Por tanto, se puede considerar que las dos vías para entregar el mensaje se han conseguido realizar de manera correcta, aunque el sistema realmente tenga más funciones de las que se planteó en la aclaración de los objetivos del trabajo. Esto es así debido a que, al principio, las funciones de guardar un rostro, buscar un rostro, listar todos los rostros, eliminar un rostro y eliminar todos los rostros no eran funciones que se plantearan realizar para el sistema, sino que se planteaba que los rostros de los usuarios que quieran utilizar el sistema estuvieran ya guardados en el NAO como requisito previo a la utilización del sistema. Esta decisión se cambió con el objetivo de hacer un sistema que fuera lo más completo posible y permitiera trabajar con los rostros, aunque sólo sea el usuario administrador quien pueda realizar estas funciones.

Para finalizar, se puede concluir que no sólo se ha obtenido un sistema que cumple con los objetivos planteados al inicio del proyecto, sino que se dispone de un sistema mucho más completo que permite que sea independiente y no se tenga que trabajar nada con Python. Como excepción hay que mencionar que el usuario administrador, que se da como requisito que sepa utilizar Python y el NAO, debe guardar su rostro en el NAO previamente antes de utilizar el sistema, dado que el reconocimiento facial es la forma que finalmente se ha elegido para permitir o no el uso del sistema a los diferentes usuarios.

9.2 Conclusiones obtenidas

Este proyecto ha supuesto bastante más dificultad de la esperada, seguramente porque al inicio de este el alumno no tenía conocimiento alguno sobre la Planificación Automática, PDDL o el NAO.

Como se mencionó en el primer capítulo de este trabajo, el alumno sí que era consciente de que este proyecto tenía bastante más dificultad que uno, por ejemplo, de programación web. Sin embargo, las dificultades encontradas fueron bastante mayores de las que se esperaban.

Durante el proyecto, aparecieron diferentes dificultades, como el hecho de tener que ir hasta la universidad para poder trabajar con el NAO, la complejidad de PELEA o los problemas típicos de trabajar con un robot (desconexiones de cables, apagados repentinos...), que

hicieron que en ocasiones el alumno se llegara a plantear incluso un posible cambio de Trabajo de Fin de Grado.

Pero una vez que el proyecto ha finalizado, el alumno se siente bastante satisfecho con la elección que realizó, además de poder disfrutar de un sistema que permite al NAO realizar acciones de bastante dificultad y que siempre es bastante interesante de mostrar, al tratarse de un trabajo sobre robótica.

Con la realización de este trabajo, se han obtenido conclusiones bastante importantes:

- La Planificación Automática es una rama de la inteligencia artificial que tiene bastante más potencial del que pueda parecer que tiene a priori, permitiendo resolver problemas bastantes más complejos que el planteado en este trabajo.
- El trabajo en robótica es bastante sacrificado y complejo, conllevando en muchas ocasiones bastantes más horas de las planificadas para el desarrollo.
- La realización de la arquitectura cliente-servidor en este trabajo ha sido de gran aprendizaje para el alumno, permitiendo ver como dos lenguajes que se vieron por separado en la carrera (Java y Python) pueden interactuar entre ellos.
- Las dificultades existentes para mover a un robot, como por ejemplo un suelo que no es completamente estable o distancias concretas planificadas para recorrer que el robot no es capaz de realizar con exactitud.
- La importancia de valorar todos los aspectos legales que conciernen a un proyecto antes de su realización.
- La definición de un presupuesto y una correcta planificación que permitan a un proyecto tener un grado de viabilidad óptimo.

Por tanto, la realización de este trabajo ha supuesto una gran fuente de conocimiento para el alumno, permitiendo haber obtenido conocimientos en robótica y haber comprendido todos los aspectos clave a considerar en la realización de un proyecto.

9.3 Líneas futuras de trabajo

Como ya se ha mencionado, este proyecto ofrece diversas posibilidades que permiten ampliar su alcance y, por tanto, su complejidad.

En concreto, se han tenido en cuenta dos posibles vías para mejorar el proyecto, que van a ser explicadas mencionando y explicando los módulos del NAO que hacen posible su realización.

En primer lugar, uno de los grandes problemas que se tiene en la navegación por un entorno que se realiza en este trabajo es la posible existencia de obstáculos. Es cierto que al cubrir las caídas del NAO haciendo que se levante y continúe puede ser una solución para ciertos obstáculos. Pero es posible utilizar los dos sonar que el NAO lleva incorporados para detectar aquellos obstáculos que pudieran estar presentes, esquivándolos dichos obstáculos y continuando el recorrido.

El módulo que permite que el NAO detecte obstáculos para que los esquive es ***ALSonar*** [49]. Este módulo trabaja con los dos sonar que tiene el NAO y que le permiten detectar obstáculos tanto a la derecha como a la izquierda de él. En concreto, se pueden dar cinco eventos que determinarán el comportamiento que tiene el NAO durante su movimiento:

- *SonarLeftDetected*: el NAO detecta un obstáculo a la izquierda a menos de medio metro, por lo que el NAO no puede seguir adelante y debe detenerse para posteriormente girar a la derecha con el objetivo de evitar el obstáculo.
- *SonarRightDetected*: el NAO detecta un obstáculo a la derecha a menos de medio metro, por lo que el NAO no puede seguir adelante y debe detenerse para posteriormente girar a la izquierda con el objetivo de evitar el obstáculo.
- *SonarLeftNothingDetected*: el NAO no tiene obstáculos delante ni a su izquierda, por lo que puede continuar recto o girar a la izquierda sin importar el obstáculo presente a la derecha y que se encuentra a menos de medio metro.
- *SonarRigthNothingDetected*: el NAO no tiene obstáculos delante ni a su derecha, por lo que puede continuar recto o girar a la derecha sin importar el obstáculo presente a la izquierda y que se encuentra a menos de medio metro.
- *SonarLeftNothingDetected* y *SonarRigthNothingDetected*: el NAO no tiene obstáculos delante, ni a la derecha ni a la izquierda.

Por tanto, utilizando este módulo se puede hacer que el NAO evite los obstáculos durante su recorrido, como se puede ver en el vídeo grabado por el Grupo de Planificación y Aprendizaje de la Universidad Carlos III de Madrid [50].

En segundo lugar y último lugar, el hecho de que sean las marcas que el NAO reconoce las pistas que le guían en su recorrido en este trabajo implica tener que imprimirlas y colocarlas en los lugares oportunos. Entonces, la solución que se plantea es en vez de utilizar estas marcas para guiar el recorrido, es utilizar cualquier objeto que se desee para que el NAO lo reconozca.

El módulo que permite al NAO reconocer objetos que ha aprendido previamente es ***ALVisionRecognition*** [51]. Para comprobar si se han reconocido objetos se debe acceder mediante el módulo *ALMemory* a la clave de memoria *PictureDetected*, que tiene dos campos. El primer campo es *TimeStamp* y es la marca de tiempo de la imagen que se utilizó para realizar la detección. El segundo campo es *PictureInfo* y dentro de él habrá tantos campos como objetos esté reconociendo el NAO, pudiendo obtener información de cada uno de ellos.

El proceso de aprendizaje de objetos se realiza mediante el monitor de vídeo de Choregraphe [52], en el que sólo hay que colocar el objeto delante del NAO para poder empezar su aprendizaje.

BIBLIOGRAFÍA

[1] Xataka Ciencia, «El origen de la palabra robot,» [En línea]. Available: https://www.xatakaciencia.com/robotica/el-origen-de-la-palabra-robot. [Último acceso: 17 junio 2019].

[2] Nexus Robótica, «Elektro, el primer robot de la historia,» [En línea]. Available: http://nexusrobotica.com/elektro-el-primer-robot-de-la-historia/. [Último acceso: 17 junio 2019].

[3] HondaDreams, «Las caras de ASIMO,» [En línea]. Available: https://www.hondadreams.es/2014/07/22/las-caras-de-asimo/. [Último acceso: 17 junio 2019].

[4] Abadía Digital, «Historia de los robots de Honda: del proyecto E0 a ASIMO,» [En línea]. Available: https://www.abadiadigital.com/historia-de-los-robots-de-honda-del-proyecto-e0-a-asimo/. [Último acceso: 17 junio 2019].

[5] Blogger.com, «HISTORIA DEL ARTE DE LA ROBÓTICA: ROBOTS HUMANOIDES,» [En línea]. Available: http://robotik-jjlg.blogspot.com/2009/06/robots-humanoides.html. [Último acceso: 17 junio 2019].

[6] RTVE, «ASIMO, 10 años cumpliendo la ley de la robótica,» [En línea]. Available: http://www.rtve.es/noticias/20101016/asimo-diez-anos-cumpliendo-leyes-robotica/362253.shtml. [Último acceso: 17 junio 2019].

[7] ResearchGate, «Características técnicas del robot NAO. (Aldebaran Robotics, 2010).,» [En línea]. Available: https://www.researchgate.net/figure/Figura-12-Caracteristicas-tecnicas-del-robot-NAO-Aldebaran-Robotics-2010_fig2_268746492. [Último acceso: 17 junio 2019].

[8] Agencia EFE, «Hiroshi Ishiguro: Viviremos en una sociedad de robots en un par de años,» [En línea]. Available: https://www.efe.com/efe/america/tecnologia/hiroshi-ishiguro-viviremos-en-una-sociedad-de-robots-un-par-anos/20000036-3167132. [Último acceso: 17 junio 2019].

[9] El Mundo, «El robot Atlas: una 'nueva especie' de humanoide para misiones peligrosas,» [En línea]. Available: https://www.elmundo.es/ciencia/2016/02/24/56cd888746163f181f8b4633.html. [Último acceso: 17 junio 2019].

[10] Boston Dynamics, «Atlas, The Next Generation,» [En línea]. Available: https://www.youtube.com/watch?v=rVlhMGQgDkY. [Último acceso: 17 junio 2019].

[11] Semantic Scholar, «Automated Planning and Scheduling for Planetary Rover Distributed Operations,» [En línea]. Available: https://pdfs.semanticscholar.org/bfa2/7d89794a07f6c09a636f074569cdc960be92.pdf.

[Último acceso: 17 junio 2019].

[12] Department of Computer and Information Science (IDA), «Writing Planning Domains and Problems in PDDL,» [En línea]. Available: https://www.ida.liu.se/~TDDC17/info/labs/planning/2004/writing.html. [Último acceso: 17 junio 2019].

[13] Planning.Domains, «PDDL Editor,» [En línea]. Available: http://editor.planning.domains/. [Último acceso: 17 junio 2019].

[14] University of Toronto Computer Science, «An Introduction to PDDL,» [En línea]. Available: http://www.cs.toronto.edu/~sheila/2542/w09/A1/introtopddl2.pdf. [Último acceso: 17 junio 2019].

[15] Planning & Learning research Group UC3M, «Introducción a PELEA,» [En línea]. Available: http://www.plg.inf.uc3m.es/pelea/index.php. [Último acceso: 17 junio 2019].

[16] FedEx, «Delivering the Future: FedEx Unveils Autonomous Delivery Robot,» [En línea]. Available: https://about.van.fedex.com/newsroom/thefuturefedex/. [Último acceso: 17 junio 2019].

[17] FedEx, «Meet the FedEx SameDay Bot,» [En línea]. Available: https://www.youtube.com/watch?v=N0rt_HB7vd4. [Último acceso: 17 junio 2019].

[18] Prestazion, «Roomba 981: descubre por qué te puede merecer la pena pagar su precio,» [En línea]. Available: https://hogar.prestazion.com/roomba-980-981.html. [Último acceso: 17 junio 2019].

[19] By AnnaM, «iRobot Roomba 981,» [En línea]. Available: https://www.youtube.com/watch?v=stZergiP_fc. [Último acceso: 17 junio 2019].

[20] ATRIA Innovation, «AVGs, ¿Vehículos de Guiado Automático todavía no sabes lo que son?,» [En línea]. Available: http://atriainnovation.com/agvs-vehiculos-de-guiado-automatico-todavia-no-sabes-lo-que-son/#more-4165. [Último acceso: 17 junio 2019].

[21] Jungheinrich AG, «Automated Guided Vehicles (English),» [En línea]. Available: https://www.youtube.com/watch?v=mEzCMS50mtE. [Último acceso: 16 abril 2019].

[22] FPT Software, «{FPT Software} Pepper Robot - first recepcionist robot at FPT,» [En línea]. Available: https://www.youtube.com/watch?v=LT4G161ImqE. [Último acceso: 17 junio 2019].

[23] Ever AI, «Ever AI,» [En línea]. Available: https://ever.ai/. [Último acceso: 17 junio 2019].

[24] FindBiometrics, «Facial Recognition Upgrade Improves Robot´s People Skills,» [En línea]. Available: https://findbiometrics.com/facial-recognition-robot-people-skills-505026/. [Último acceso: 17 junio 2019].

[25] SmartCitiesWorld, «Robots equipped with facial and voice recognition at MWC19,» [En línea]. Available: https://www.smartcitiesworld.net/news/news/robots-equipped-with-facial-and-voice-recognition-at-mwc19-3889. [Último acceso: 17 junio 2019].

[26] New Era AI Robotic, «Smart Service Robot - Office Building/Retail Store/Restaurant,» [En línea]. Available: https://www.youtube.com/watch?v=t8jIqxQZAiw. [Último acceso: 17 junio 2019].

[27] My Robelf, «Robelf 2016 Commercial Full Version,» [En línea]. Available: https://www.youtube.com/watch?v=8Yyv_NFvuUs. [Último acceso: 17 junio 2019].

[28] Peru.com, «MWC 2019: los 5 robots que roban miradas en el Congreso de Móviles de Barcelona,» [En línea]. Available: https://peru.com/epic/epic-mobile/mwc-2019-5-robots-que-roban-miradas-congreso-moviles-barcelona-fotos-5g-viral-noticia-598313. [Último acceso: 17 junio 2019].

[29] Arirang TV, «5G Robot baristas, and smart technologies bringing change the South Korean lifestyle,» [En línea]. Available: https://www.youtube.com/watch?v=atZ6AzHhBpk. [Último acceso: 17 junio 2019].

[30] Nuwa Robotics, «Kebbi Robot AI,» [En línea]. Available: https://www.youtube.com/watch?v=XGkl7kdPZa0. [Último acceso: 17 junio 2019].

[31] W3Schools, «What is Client-Server Architecture?,» [En línea]. Available: https://www.w3schools.in/what-is-client-server-architecture/. [Último acceso: 17 junio 2019].

[32] Aldebaran Robotics, «ALTextToSpeech,» [En línea]. Available: http://doc.aldebaran.com/2-1/naoqi/audio/altexttospeech.html. [Último acceso: 17 junio 2019].

[33] Aldebaran Robotics, «ALSpeechRecognition,» [En línea]. Available: http://doc.aldebaran.com/2-1/naoqi/audio/alspeechrecognition.html. [Último acceso: 17 junio 2019].

[34] Aldebaran Robotics, «ALMemory,» [En línea]. Available: http://doc.aldebaran.com/2-1/naoqi/core/almemory.html. [Último acceso: 17 junio 2019].

[35] Aldebaran Robotics, «ALAudioRecorder,» [En línea]. Available: http://doc.aldebaran.com/2-1/naoqi/audio/alaudiorecorder.html. [Último acceso: 17 junio 2019].

[36] Aldebaran Robotics, «ALRobotPosture,» [En línea]. Available: http://doc.aldebaran.com/2-1/naoqi/motion/alrobotposture.html. [Último acceso: 17 abril 2019].

[37] Aldebaran Robotics, «ALMotion,» [En línea]. Available: http://doc.aldebaran.com/2-1/naoqi/motion/almotion.html. [Último acceso: 17 junio 2019].

[38] Aldebaran Robotics, «ALTracker,» [En línea]. Available: http://doc.aldebaran.com/2-1/naoqi/trackers/altracker.html. [Último acceso: 17 junio 2019].

[39] Aldebaran Robotics, «ALPeoplePerception,» [En línea]. Available: http://doc.aldebaran.com/2-1/naoqi/peopleperception/alpeopleperception.html. [Último acceso: 17 junio 2019].

[40] Aldebaran Robotics, «ALFaceDetection,» [En línea]. Available: http://doc.aldebaran.com/2-1/naoqi/peopleperception/alfacedetection.html. [Último acceso: 17 junio 2019].

[41] Aldebaran Robotics, «ALAudioPlayer,» [En línea]. Available: http://doc.aldebaran.com/2-1/naoqi/audio/alaudioplayer.html. [Último acceso: 17 junio 2019].

[42] Aldebaran Robotics, «Python SDK,» [En línea]. Available: http://doc.aldebaran.com/2-1/dev/python/index.html. [Último acceso: 17 junio 2019].

[43] BlogThinkBig.com, «Qué dicen las tres leyes de la robótica de Isaac Asimov,» [En línea]. Available: https://blogthinkbig.com/que-dicen-las-tres-leyes-de-la-robotica-de-isaac-asimov. [Último acceso: 17 junio 2019].

[44] Parlamento Europeo, «Proyecto de informe con recomendaciones destinadas a la Comisión sobre normas de Derecho civil sobre robótica,» [En línea]. Available: http://www.europarl.europa.eu/sides/getDoc.do?pubRef=-//EP//NONSGML%2BCOMPARL%2BPE-582.443%2B01%2BDOC%2BPDF%2BV0//ES. [Último acceso: 17 junio 2019].

[45] BlogThinkBig.com, «Las 6 leyes de la robótica de la Unión Europea,» [En línea]. Available: https://blogthinkbig.com/las-6-leyes-de-la-robotica-de-la-union-europea. [Último acceso: 17 junio 2019].

[46] Agencia Estatal Boletín Oficial del Estado, «Ley Orgánica 3/2018, de 5 de diciembre, de Protección de Datos Personales y garantía de los derechos digitales,» [En línea]. Available: https://www.boe.es/boe/dias/2018/12/06/pdfs/BOE-A-2018-16673.pdf. [Último acceso: 17 junio 2019].

[47] InfoJobs, «Estado del mercado laboral en España,» [En línea]. Available: https://nosotros.infojobs.net/wp-content/uploads/2018/05/Informe-Anual-InfoJobs-ESADE-2017-Completo.pdf. [Último acceso: 17 junio 2019].

[48] Cuestiones laborales, «Cómo calcular la jornada de trabajo,» [En línea]. Available: https://www.cuestioneslaborales.es/como-calcular-la-jornada-de-trabajo/. [Último acceso: 17 junio 2019].

[49] Aldebaran Robotics, «ALSonar,» [En línea]. Available: http://doc.aldebaran.com/2-1/naoqi/sensors/alsonar.html. [Último acceso: 17 junio 2019].

[50] Planning & Learning research Group UC3M, «NAO avoiding an obstacle using sonar

sensor,» [En línea]. Available: https://www.youtube.com/watch?v=_YIX5beZZMA. [Último acceso: 17 junio 2019].

[51] Aldebaran Robotics, «ALVisionRecognition,» [En línea]. Available: http://doc.aldebaran.com/2-1/naoqi/vision/alvisionrecognition.html. [Último acceso: 17 junio 2019].

[52] Aldebaran Robotics, «Recognizing objects,» [En línea]. Available: http://doc.aldebaran.com/2-1/software/choregraphe/tutos/recognize_objects.html. [Último acceso: 17 junio 2019].

ANEXO A: RESUMEN EN INGLÉS

ABSTRACT

This work corresponds to the development of a message exchange system that will allow a sender user to deliver a message to a receiver user by two ways. The first way, if the message is personal, will be to add the message as an audio attachment to an email that will be sent to the receiver user's email. The second way, if the message is not personal, will be that the message will be delivered by the NAO robot, where the NAO will make a route to reach the receiver user and will recognize this user through facial recognition. Finally, if the second way is used, the receiver of the message will be able to send a reply that can be delivered by either of the two ways mentioned above.

1. INTRODUCTION

Why do robots exist? The answer to this question is based on the human need to create machines capable of carrying out people's own tasks. In this way, robots could be assigned tasks that are repetitive or pose a risk to humans. But, what if it could be done that the robot will perform more complex tasks or react to adverse situations? Two concepts that allow this fact are Artificial Intelligence (computer field that allows the creation of intelligent machines capable of performing complex tasks) and Automated Planning (branch of Artificial Intelligence that allows a robot to create an actions plan that solves a problem and have the ability to make a new plan if the conditions of the problem change).

In this way, it is intended to use the NAO robot by means of Automated Planning to create a message exchange system that will allow two users to exchange messages between them either by delivering the message by email or by delivering the message by the NAO robot to the receiver user.

The motivation for this work is based on the creation of a powerful system that has as main novelty the joint use of Automated Planning, navigation in an environment and facial recognition. In addition, it represents a great formation for the student, given that he realized in the degree the branch of Information Systems.

The document is structured in nine chapters. The first chapter is the introduction and offers a short description about the work, as well as showing the motivation for doing the work and the structure of the document. The second chapter is the state of the art and involves a review

of the technologies used in this work and their current state. The third chapter is the objectives of the work and describes the objectives to be achieved with the project. The fourth chapter is the analysis of the system and describes all the functionalities that the system must have. The fifth chapter is the design and implementation of the system and explains the architecture of the system to be built and the implementation that complies with that architecture. The sixth chapter is the tests of the system and describes the tests that have been carried out to verify that the system fulfills all the necessary functions. The seventh chapter is the regulatory framework and conducts a study of the legal restrictions that apply to the project, names the licenses of the tools used and the standards used and clarifies the aspects related to the intellectual property of the work. The eighth chapter is the socioeconomic environment and carries out project planning and budget, as well as an analysis of the socioeconomic impact of the system developed. The ninth and final chapter contains the conclusions and future work and explains the objectives reached, the conclusions obtained and the possible future lines of the work.

2. STATE OF THE ART

As mentioned in the previous chapter, this work aims to use the NAO robot, through Automated Planning (with the use of the PELEA architecture), to build a system that allows the exchange of messages between two users, using facial recognition and navigation in an environment.

But what is a robot? A robot is a programmable machine capable of carrying out actions that before could only be carried out by humans. The first robots, such as the **Elektro** robot (1937), were only able to perform static actions such as talking and moving the head and arms. Over the years, walking robots emerged, such as Honda's **E0** and **E6** models, even though they only had the lower part of a robot. The next step for Honda involved the creation of complete robots (with the whole body) capable of moving, such as the **P1** and **ASIMO** robots.

After this, robots with multiple functionalities appeared, such as the **NAO** robot, capable of moving, speaking, recording and reproducing audios or recognizing objects and people. In addition, robots focused on movement in environments with irregular ground (**Atlas**) or robots focused on having an appearance almost equal to that of a human being (**Geminoid HI 4**) appeared, the latter being even able to give conferences.

Knowing what a robot is and what it implies, the aim is to give it autonomy so that it does not only carry out repetitive tasks. This is possible thanks to the Automated Planning, which is a discipline of Artificial Intelligence whose objective is the production of plans that allow to reach certain goals.

It is a discipline with quite interesting applications (such as the use of **rovers** in space missions on Mars), and is applied by automated planners, which are programs that incorporate specific algorithms of Automated Planning.

These planners receive the task to be solved expressed through planning languages, which allow to express all the conditions of the task to be solved. The language used in this work is PDDL (Planning Domain Definition Language), since it is an attempt to standardize the planning languages.

The use of Automated Planning and PDDL is reflected in the use of the PELEA architecture (Planning, Execution and Learning Architecture), which is an open-source architecture that allows planning, execution, monitoring, replanning and learning tasks oriented to robot control developed by the Carlos III University of Madrid, the Polytechnic University of Valencia and the University of Granada.

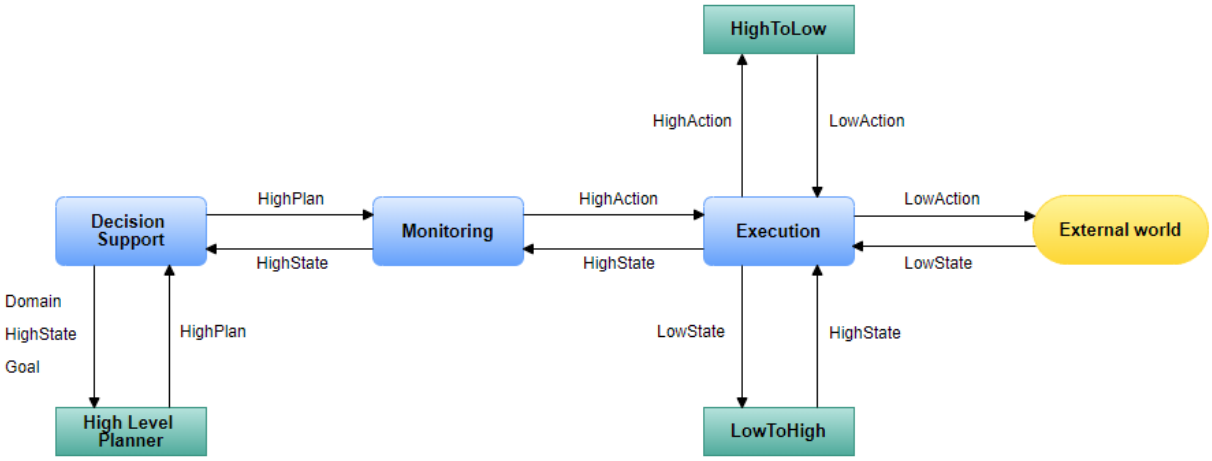
In it, the actions to be carried out have two levels: high level (actions generated by the automated planner) and low level (actions carried out by the robot).

Its functioning is based on the interaction of three modules: Decision Support, Monitoring and Execution. Decision Support is the module that receives the task to be solved and generates a plan of actions to be carried out thanks to the use of an automated planner, generating a new plan in case replanning is necessary.

Monitoring receives from Decision Support the plan to be carried out and sends to the Execution module each one of the high-level actions to be carried out. When sending a high level action, it receives a high level state of the problem. If the received state is the expected one, the next action is sent to Execution and if it is not, the state of the problem is sent to the Decision Support module to generate a new plan.

Finally, the Execution module is in charge of translating the high level action received to low level with the HighToLow component. Then, the low-level action would be sent to the robot to perform it. Once the action had been performed, the Execution module would receive the

low level status of the problem. Once received, it would translate the high-level status with the LowToHigh component and send it to the Monitoring module:



Considering navigation in a robotic environment, two interesting examples are **FedEx SameDay Bot**, capable of short-distance package delivery, and **Roomba 981**, which is a smart vacuum cleaner capable of cleaning a house without repeating areas it has already cleaned.

Considering the recognition of people using robots, there are different models of robots that do it satisfactorily:

- **Pepper:** able to recognise people, being able to detect if the detected face is from a photograph and is real, obtain the age, gender or ethnic origin of a person and capture the type of emotions of a person.
- **Smart Service:** interlocutor in shops and offices to advise the user.
- **Robelf:** able to recognise family members and strangers entering the home.
- **Barista:** able to recognise users and serve them a personalized coffee.
- **Kebbi:** able to take care of children and recognize possible strangers entering the house.

3. WORK OBJECTIVES

The objective of this work is to create a system that allows two modes of delivery of a message. If the message is personal, it will be delivered as an audio attachment in an email that will be sent to the email of the receiver user. If the message is not personal, the message will be delivered by the NAO, carrying out a route to reach the receiver user, recognizing this user through facial recognition and delivering the message. In this last delivery mode, a reply by the receiver of the message will be allowed, sending this reply by any of the two available ways.

The achievement of a system that satisfies these characteristics is composed of the following phases:

- Study and experimentation with NAO.
- Study of PELEA.
- Study of PDDL.
- Definition of the domain and the problem to be solved in this work with PDDL.
- Necessary modifications of PELEA.
- Creation and implementation of a server that receives an action of PELEA, executes it in the NAO and returns to PELEA information about the status of the action.
- Tests that verify the correct functioning of the system.

Once all the phases have been carried out, it is intended to have a functional system that allows adding complexity to the problem to be solved without having to make too many modifications.

4. SYSTEM ANALYSIS

The system to be built must have, in order to be more complete, apart from the message exchange functions mentioned in the work objectives, a series of functions that allow for the management of faces stored inside the NAO robot.

Therefore, the system must have the following functions:

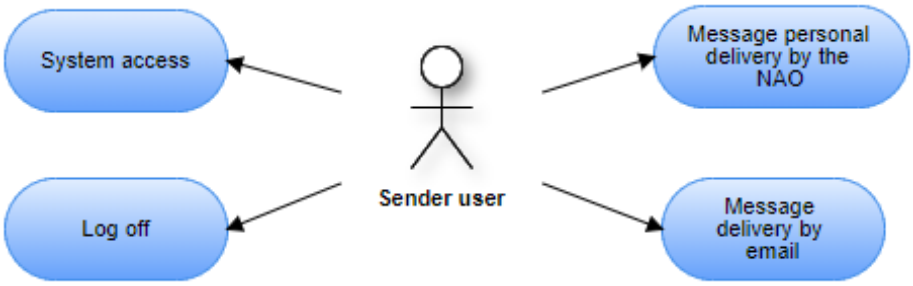
- Learn a face.
- Search for face.
- List all faces.
- Remove a face.
- Remove all faces.

- Delivery of message from sender user to receiver user by email.
- Personal delivery of message from sender user to receiver user.
- Delivery of message from receiver user to sender user by email.
- Personal delivery of message from receiver user to sender user.

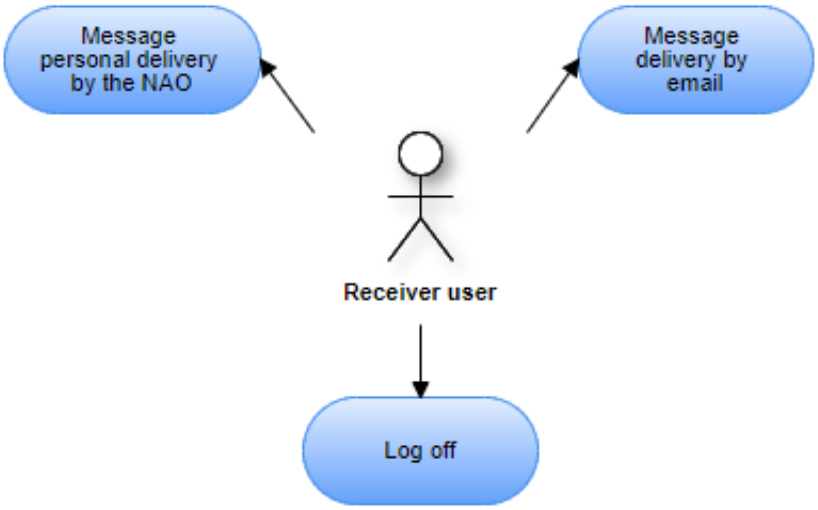
For security reasons, a users file will be used to determine which users are authorized to access the system and what role they have, whether normal or administrator. The first five functions (face management) can only be performed by the system administrator user.

In this way, the three actors that will interact in the use cases are identified: sender user, receiver user and administrator.

The use cases that the sender user will be able to make will be:



The use cases that the receiver user will be able to perform will be:



Finally, the use cases that the administrator will be able to perform will be:

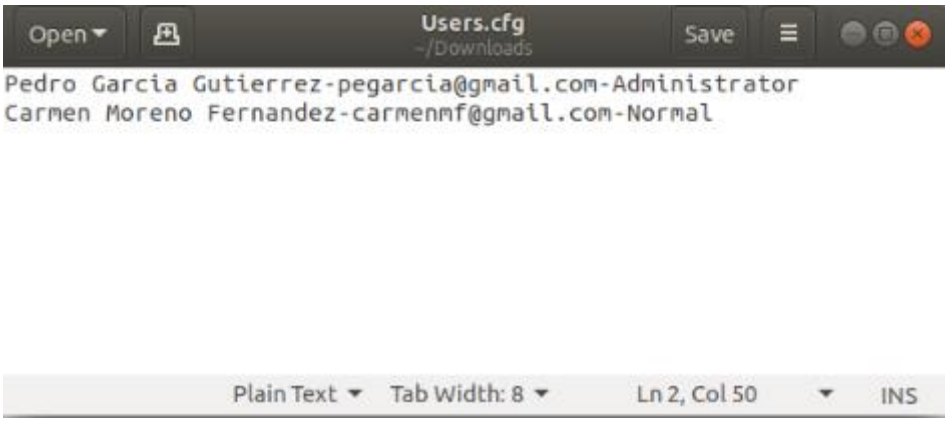
Therefore, the use cases that the system will have will be:

- Configuration of authorized users to use the system.
- Configuration of the problem in PDDL.
- System access.
- Face learning.
- Face search.
- All faces list.
- Face elimination.
- All faces elimination.
- Delivery of a message from the sender user by e-mail.
- Physical delivery of a message from the sender user.
- Delivery of a message from the receiver user by e-mail.
- Physical delivery of a message from the receiver user.
- Log off.

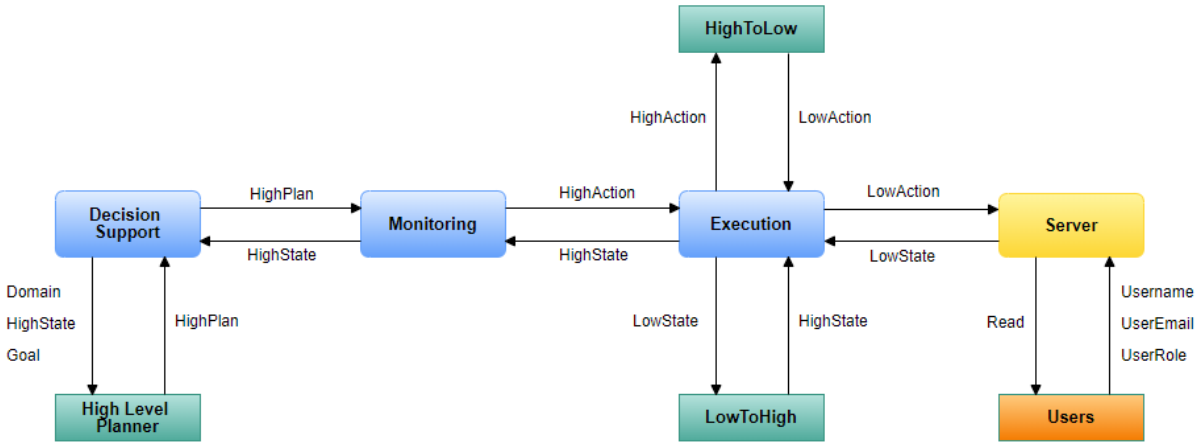
5. SYSTEM DESIGN AND IMPLEMENTATION

The architecture of the system to be developed will be the same as that of the PELEA architecture, to which will be coupled a server programmed in Python that will receive a low level action of PELEA, will execute it in the NAO and will return to PELEA the state of the action carried out.

This server will work with the users of the system through the interaction with an authorized users file, in which in each one of its lines it will put the name of a user, a delimiter (-), its email, another delimiter (-) and its role in the system:



In this way, the final architecture of the system would be:



In order to perform the functions that the system allows, seventeen actions will be available, each of which will have a high-level version and another low-level version. The high-level version of the actions will be stored in the PDDL file containing the domain of the problem and the low-level version will be found on the Python server.

In this way, there are the following actions:

- **getFunction:** gets the function to be performed.
- **learnFace:** stores a face in the NAO.
- **searchFace:** searches a face in the NAO.
- **listFaces:** lists all faces stored in the NAO.
- **removeFace:** removes a face from the NAO.
- **clearDatabase:** removes all faces stored in the NAO.
- **getInformation:** gets information about the user who must receive the message.
- **recordMessage:** records the message of the sender user.
- **typeMessage:** obtains if the message to be delivered is personal or not.
- **sendMessage:** sends the message by email.
- **findMark:** searches a mark of the route to be performed by the NAO.
- **goToMark:** makes the NAO reach a certain found mark.
- **checkGoalMark:** reports that the last mark of the route has been reached.
- **findPerson:** searchs for a person.
- **recognisePerson:** checks if the person found is the receiver user.
- **playMessage:** reproduces the message to the receiver user.
- **rebuildRoute:** reconstructs the route to deliver a response to the message.

Depending on the function of the system being carried out, one action or another will be executed:

Function	Actions
Learn a face	*getFunction* and *learnFace*
Search a face	*getFunction* and *searchFace*
List all faces	*getFunction* and *listFaces*
Remove a face	*getFunction* and *removeFace*
Remove all faces	*getFunction* and *clearDatabase*
Delivery of message by email	*getFunction*, *getInformation*, *recordMessage*, *typeMessage* and *sendMessage*

Delivery of message by NAO itself	*getFunction*, *getInformation*, *recordMessage*, *typeMessage*, *findMark*, *goToMark*, *checkGoalMark*, *findPerson*, *recognisePerson*, *playMessage* and *rebuildRoute** (If the user wants to reply to the received message)
Log off	*getFunction*

6. TESTS

This chapter explains the tests carried out on the system to verify that it fulfils all the required functions. They have been carried out in laboratory 2.1.B16 of the Sabatini Building belonging to the Carlos III University of Madrid, on the Leganés campus.

The first group of tests focused on user's access to the system, allowing it only to users with the role of normal user or administrator and denying access to users without a saved face, to non-authorized users even with a saved face and to users with the incorrect format of their email, also checking that if there are no users in the file of authorized users, no one is allowed access to the system.

The second group of tests focused on checking the correct functioning of functions related to faces and their corresponding cases of error: a repeated face (add), a face of an unauthorized user (add), no faces stored in the NAO database (search, list, delete a face and delete all faces) or a non-existent face (delete a face).

The third group of tests focused on checking the correct functioning of exiting the system and delivering a message, both in the case of sending it by mail and in the case of delivery by the NAO (also checking in this case the delivery of a response from the receiver user).

The rest of the tests carried out have focused on verifying that the system repeats a question in the case that the user answers it incorrectly (function to be performed, duration of the message, type of message and response to the message received), that the receiver user complies with certain restrictions (he is authorised, the format of his mail is correct and has his face saved in the case that the NAO delivers the message) and that the navigation is correct (relocation of marks in the case of loss, search for people and correct identification of the receiver user).

All the tests carried out have had the expected and correct behaviour.

7. REGULATORY FRAMEWORK

This chapter explains the legal restrictions that may affect the project, the licenses of the tools used, the technical standards used, and issues related to the intellectual property of the work.

There are two approaches to legal restrictions. First, since this is robotic work, the regulatory framework for robots must be taken into account. Since there is currently no legislation regulating the use of robots, a resolution adopted by the European Parliament is mentioned in which the European Union is urged to develop specific legislation on the use of Artificial Intelligence and its limits.

Secondly, given that we deal with data on people in this work, it is necessary to take into account *Ley Orgánica 3/2018, de 5 de diciembre, de Protección de Datos Personales y garantía de los derechos digitales*. Therefore, all the data used in this work will comply with this law.

With regard to the tools used in this work, each of them is under a specific software license:

- **Microsoft Products:** Both the version of Microsoft Word and the version of Microsoft Project used in this work are licensed for the educational field.
- **Java:** is under the GNU General Public License.
- **Eclipse:** is under the Eclipse Public License.
- **Python:** is under the Python Software Foundation License.
- **Spyder:** is under an MIT (Massachusetts Institute of Technology) license.
- **GnuPG:** is under the GNU General Public License.
- **Pelea:** is under a Creative Commons license.
- **Choregraphe:** is under an EULA (End User License Agreement) license.

With regard to the technical standards used in the work, the only one used is the following:

- **IEEE format:** has been used for the realization of the references of this work.

Finally, with regard to intellectual property, the work is oriented to research, so it will have a Creative Commons license that will allow any of the students or faculty of the University Carlos III of Madrid to use it for free.

8. SOCIAL AND ECONOMIC CONTEXT

This chapter explains the planning of the project, the budget associated to the work and the expected socioeconomic impact of the project.

The planning of the project has the following phases: meeting with the tutor for the selection of the work, meeting with the tutor for the specification of objectives, state of the art, study of the regulatory framework, system analysis, system design and implementation, system testing, conclusions and documentation of the project. The estimated start date of the project is 21 November 2018 and the estimated end date is 31 May 2019. Due to external problems to the project and to the fact that some phases suffered modifications in their duration, the real start date of the project was 3 December 2018 and the real end date was 17 June 2019.

With regard to the budget section, the cost of staff, the cost of equipment, the cost of consumables, the cost of travel and daily subsistence allowance and indirect costs have been taken into account, resulting in a total cost of **14,133.46 €**. A ten percent risk rate, twenty percent profit rate and twenty-one percent tax rate have been applied to the previous cost, obtaining a final cost of **22,573.97 €**.

With regard to the socioeconomic section, economic impact, social impact and ethical impact have been taken into account. The environmental impact has not been taken into account because it does not apply to this project.

In the economic aspect, it is not expected to obtain any benefit from the work, since it is oriented to research purposes. In the social aspect, a positive impact is expected because this work can facilitate the delivery of messages to people with reduced mobility. Finally, in the ethical aspect, all data used in the project will be used in accordance with *Ley Orgánica 3/2018, de 5 de diciembre, de Protección de Datos Personales y garantía de los derechos digitales*.

9. CONCLUSIONS AND FUTURE WORKS

This chapter explains the objectives achieved in the work, the conclusions obtained with its realization and the possible future lines of work.

The objective that was set at the beginning of the project was to have a system that could deliver a message by two ways. The first way would be used in case the message was personal and would consist of sending the message in an audio attachment of an email that would be sent to the email of the receiver user. The second way would be used in case the message was not personal and would consist of the NAO following a route until reaching the receiver user, recognizing him/her through facial recognition and delivering the message to

them. In this second way, it would also be possible to reply to the received message, sending the response by any of the two existing ways.

Both types of delivery have been achieved in this work, in addition to being able to include functions related to the management of faces (add, search, list and remove) in order to achieve a much more complete system.

This work has had many difficulties during its realization, but it has allowed the student to obtain a series of conclusions:

- The Automated Planning allows to solve more complicated problems than the one raised in this work.
- The implementation in robotics takes many more hours than planned.
- The client-server architecture has allowed the student to see the interaction between different programming languages.
- The existing difficulties in the navigation of the NAO.
- The importance of the evaluation of the legal aspects of a project before its realization.
- The realization of a budget and a planning that grant viability to a project.

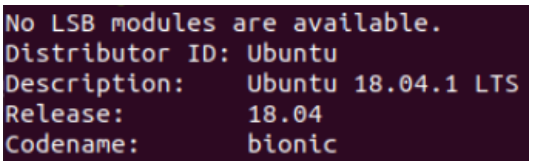
With regard to the possible future lines of work offered by this project, two possible improvements have been proposed to increase its scope and complexity:

- **Avoiding obstacles:** the current implementation covers the possible falls of the NAO during the travel of its route of marks, but it does not take into account the possible appearance of obstacles during its realization. The NAO ***ALSonar*** module makes it possible to use the two sonars of the NAO to detect obstacles during movement.
- **Learning objects:** the route followed by the NAO is made up of marks that this robot recognises and this causes it to depend on them for navigation. Choregraphe's video monitor allows the NAO to learn different objects, thus enabling that the route followed by NAO could be formed of these objects rather than recognisable marks. The NAO module that allows recognition of previously learned objects is ***ALVisionRecognition***.

ANEXO B: MANUAL DE INSTALACIÓN

1. Sistema operativo y versión

El sistema operativo que se utiliza en este manual de instalación es Ubuntu y su versión es la 18.04, como se puede comprobar en la siguiente imagen:



Para poder comprobar el sistema operativo que se tiene y su versión, tan sólo hay que ejecutar en la consola del sistema operativo, siempre que sea un sistema operativo de Linux, el siguiente comando:

```
> lsb_release -a
```

Los pasos seguidos en este manual son válidos para un sistema operativo Linux de 64 bits.

2. Creación de cuenta

Para poder instalar Choregraphe, será necesario crearse una cuenta en la página oficial de *SoftBank Robotics*[1]. La creación de la cuenta es totalmente gratuita.

3. Descarga e instalación de Choregraphe

Una vez que la cuenta ya ha sido creada, se deberá acceder a la página donde están disponibles las distintas versiones de Choregraphe para descargar[2], introduciendo el correo y la contraseña de la cuenta creada para poder acceder.

Entonces, en dicha página, se encontrará la siguiente sección:

[1] https://accounts.softbankrobotics.com/#/create

[2] https://community.ald.softbankrobotics.com/en/resources/software/former-nao-versions-choregraphe-suite

En dicha sección, se deberá descargar la opción *Choregraphe 2.1.4 Linux 64 Binaries*. Con ello, se descargará el archivo *choregraphe-suite-2.1.4.13-linux64.tar.gz*.

Una vez que el archivo ya se haya descargado, habrá que colocarlo en el directorio donde se quiera que esté instalado. Después, habrá que acceder por consola al directorio donde esté guardado y descomprimirlo con el siguiente comando:

```
> tar -xvzf choregraphe-suite-2.1.4.13-linux64.tar.gz
```

A continuación, se accederá por consola a la carpeta *etc* del sistema operativo y se realizará el siguiente comando para editar el fichero *bash.bashrc*:

```
> sudo gedit bash.bashrc
```

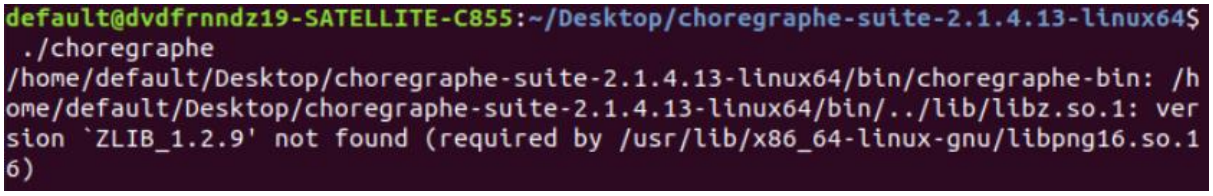
Esta edición consistirá en añadir al final de dicho fichero las siguientes variables de entorno:

```
export AL_DIR=~/Desktop/choregraphe-suite-2.1.4.13-linux64
export LD_LIBRARY_PATH=$AL_DIR/lib:$LD_LIBRARY_PATH
export PYTHONPATH=$AL_DIR/lib:$PYTHONPATH
export PATH=$PYTHONPATH:$AL_DIR/bin:$PATH
```

Estas líneas son válidas en el caso de que el directorio *choregraphe-suite-2.1.4.13-linux64* esté en el escritorio del sistema operativo (*Desktop*). En caso de que, por ejemplo, dicho fichero se encuentre en la carpeta de descargas (*Downloads*), habría que sustituir en la primera línea *Desktop* por *Downloads*.

Una vez que se haya terminado con la edición del fichero, se deberá, en primer lugar, guardar los cambios y, en segundo lugar, reiniciar el sistema operativo para que los cambios se hagan efectivos.

Llegado a este punto, se deberá acceder al directorio *choregraphe-suite-2.1.4.13-linux64*, entrar en el directorio *lib*, y eliminar el archivo *libz.so.1*, que provoca problemas en el arranque de Choregraphe, como se puede ver en la siguiente imagen:



Con el archivo ya eliminado, se accederá por consola al directorio *choregraphe-suite-2.1.4.13-linux64* y se arrancará Choregraphe utilizando el siguiente comando:

```
> ./choregraphe
```

Seguidamente, Choregraphe se iniciará y mostrará que hay que incluir un código de licencia. Para adquirir dicho código, habrá que acceder a la misma página donde se descargó la versión de Choregraphe[3], donde el código se muestra de la siguiente forma:

File	Programming language	Download link
When you launch Choregraphe, the license key is requested. Please copy and paste the following key: 654e-4564-153c-6518-2f44-7562-206e-4c60-5f47-5f45	-	-

Cuando el código haya sido introducido, finalmente Choregraphe se iniciará completamente.

La ejecución de Choregraphe durante la utilización del sistema desarrollado no es obligatoria, pero permitirá al usuario:

- Conocer la dirección IP y el puerto en el que el NAO está actuando.
- Ver la cámara de la cabeza del NAO para poder decidir la cara que quiere poner a la hora de almacenar su rostro en la base de datos del NAO.
- Ver la cámara de la cabeza del NAO para poder ver como realiza el NAO el recorrido de las marcas hasta llegar a su destino.

4. Instalación de Python

Una vez realizada la instalación de Choregraphe, se deberá instalar Python. Para ello, se debe ejecutar por consola el siguiente comando:

```
> sudo apt install python2.7-minimal
```

Una vez instalados Choregraphe y Python, ya se podrá arrancar el servidor de Python del sistema desarrollado y Choregraphe.

5. Instalación de Eclipse

El sistema desarrollado está formado, además de por un servidor en Python, de PELEA, que es un proyecto de Java. Por tanto, será necesario instalar el JDK de Java, utilizando Eclipse como entorno para trabajar con este lenguaje.

[3] https://community.ald.softbankrobotics.com/en/resources/software/former-nao-versions-choregraphe-suite

En primer lugar, para instalar el JDK de Java, se deberá ejecutar por consola el siguiente comando:

```
> sudo apt install openjdk-11-jre-headless
```

Para comprobar que se ha instalado con éxito, se puede ejecutar el comando:

```
> java -version
```

Saliendo como resultado:

```
default@dvdfrnndz19-SATELLITE-C855:~/Desktop/choregraphe-suite-2.1.4.13-linux64$
 java -version
openjdk version "10.0.2" 2018-07-17
OpenJDK Runtime Environment (build 10.0.2+13-Ubuntu-1ubuntu0.18.04.4)
OpenJDK 64-Bit Server VM (build 10.0.2+13-Ubuntu-1ubuntu0.18.04.4, mixed mode)
```

En segundo lugar, para instalar Eclipse, se deberán ejecutar por consola los siguientes comandos:

```
> wget
http://ftp.jaist.ac.jp/pub/eclipse/technology/epp/downloads/release/oxygen/3a/eclipse-
java-oxygen-3a-linux-gtk-x86_64.tar.gz
> sudo tar -zxvf eclipse-java-oxygen-3a-linux-gtk*.tar.gz -C /usr/
```

Una vez realizados estos dos comandos, la instalación de Eclipse habrá finalizado.

6. Configuración del NAO

Para utilizar el sistema desarrollado, se necesitará un robot NAO que esté listo para ser utilizado.

NAO permite dos modos de utilización. El primer modo, y el más habitual, es utilizando un router. El segundo modo, en caso de que no se disponga de router, consiste en utilizar un cable Ethernet para realizar la conexión con el NAO. Este segundo modo sólo es recomendable utilizarlo con el sistema desarrollado en caso de que se realice una función que no implique el movimiento físico del NAO, dado que el cable debe estar conectado en todo momento tanto al NAO como al ordenador para que no se pierda la conexión.

En todo caso, el cable Ethernet será necesario en cualquiera de los dos modos dado que la configuración inicial del NAO requiere de su utilización.

La documentación de *Aldebaran Robotics* provee un manual de configuración para la conexión del NAO tanto utilizando un router como utilizando un cable Ethernet[4].

7. Instalación de GnuPG

Además del servidor y del proyecto de PELEA, también hay un fichero de usuarios que se utilizará para registrar en él los usuarios que pueden utilizar el sistema desarrollado.

Dado que es un fichero para controlar la seguridad del sistema, deberá ser proporcionado a los usuarios normales cifrado para que no puedan leer ni modificar dicho fichero.

Por este motivo, se va a utilizar GnuPg (GNU Privacy Guard), una herramienta de cifrado y firmas digitales. Para realizar su instalación, se deberán ejecutar por consola los siguientes comandos:

```
> sudo apt install python-pip
> pip install python-gnupg
```

Una vez realizados estos comandos, la instalación de GnuPG habrá finalizado, pudiéndose ya cifrar y descifrar ficheros.

8. Instalación del sistema

Para instalar el sistema desarrollado, en primer lugar, habrá que descargarse la carpeta comprimida con el código del Trabajo de Fin de Grado.

Una vez descargada dicha carpeta, se deberá descomprimir y se dispondrá de los siguientes archivos:

- *PeleaTFG.tar.gz*: carpeta comprimida que contiene el proyecto Java de PELEA.
- *Server.pyc*: servidor en Python encargado de recibir las peticiones de PELEA, ejecutar las acciones en el NAO y mandar la respuesta a PELEA.
- *Users.cfg*: fichero de configuración con los usuarios que pueden utilizar el sistema desarrollado.
- *Naomarks.pdf*: documento PDF que contiene diez marcas que el NAO reconoce y que servirán como guía para realizar una ruta física con el NAO.

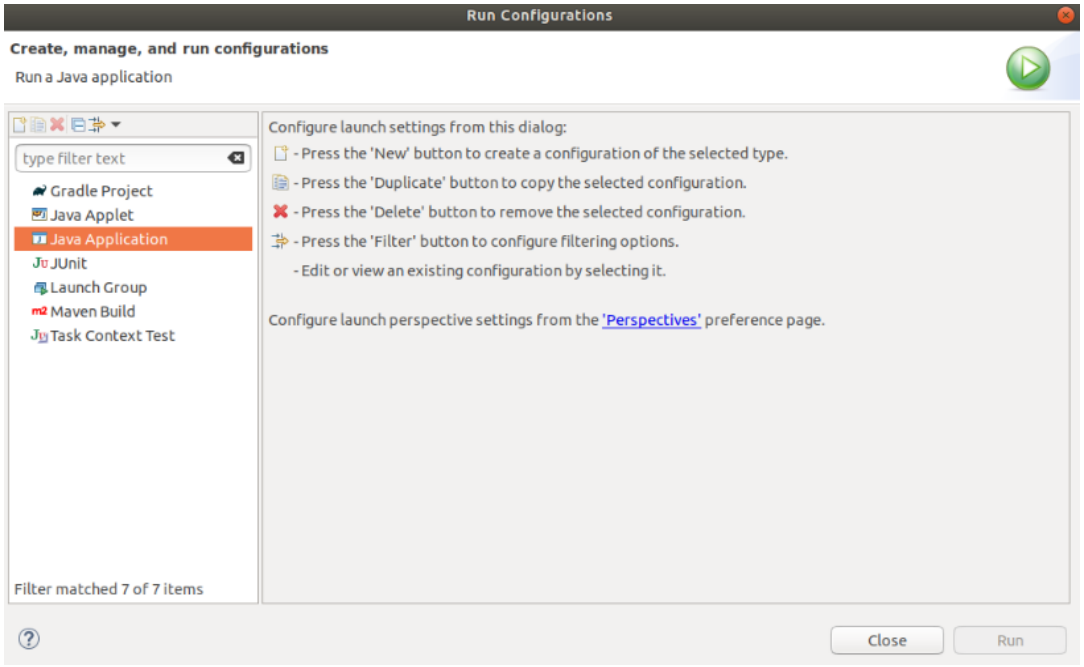
Para importar el proyecto de PELEA, se deberá iniciar Eclipse. Una vez que Eclipse esté iniciado, se deberá pulsar *File*, *Import*, *Existing Projects into Workspace*, *Next*, *Select archive*

[4] http://doc.aldebaran.com/2-1/nao/nao-connecting.html

file, *Browse* y seleccionar el archivo comprimido que contiene el proyecto de PELEA. Una vez seleccionado, se pulsará *OK* y *Finish*. Con esto, el proyecto ya estará importado.

Por último, se deberá añadir a cada una de las tres clases principales (*main.java*, *mainDS.java* y *mainEXE.java*), que representan a los módulos de *Monitoring*, *Decision Support* y *Execution* y que están dentro de la carpeta *src* y del paquete *org.pelea*, los argumentos que deben recibir.

Para ello, se pulsa en *Run* y *Run Configurations*, apareciendo la siguiente ventana:



En ella, se deberá:

- Pulsar con el botón secundario del ratón en *Java Aplication* y pulsar sobre *New*.
- En la sección que aparezca a la derecha, en la pestaña de *Main*, se debe añadir en el apartado de *Project* el proyecto de PELEA y en el apartado de *Main class* la clase Java que se corresponda con la clase principal que se está configurando (*main.java*, *mainDS.java* o *mainEXE.java*).
- En la pestaña de *Arguments*, se debe añadir en el apartado *Program arguments* la línea de argumentos que corresponda:
    - *main.java*: -c ./configs/configurationDeclarative.xml -n M1
    - *mainDS.java*: -c ./configs/configurationDeclarative.xml -n DS1
    - *mainEXE.java*: -c ./configs/configurationDeclarative.xml -n EXE1

Se deberán realizar estos tres pasos tres veces, una por cada una de las clases principales del proyecto.

Una vez realizados todos los pasos, habrá finalizado la instalación del sistema.

9. Configuración de usuarios

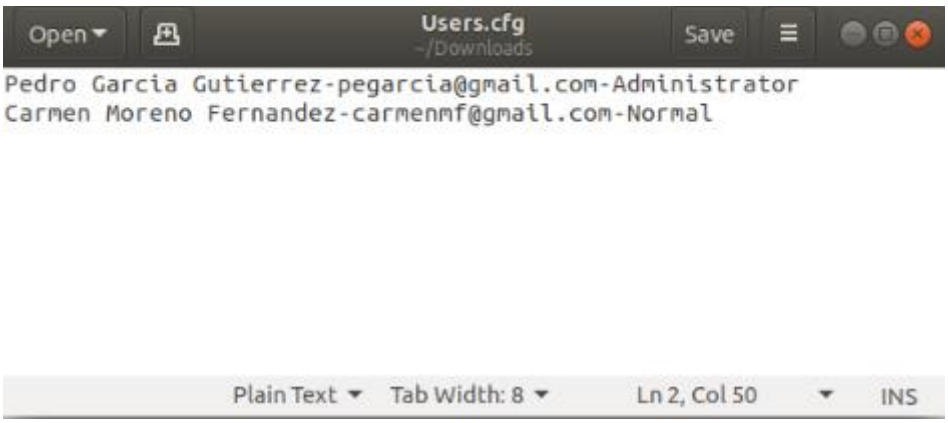
Cuando se haya realizado la instalación del sistema desarrollado, para utilizarlo habrá que editar el fichero de configuración con los usuarios que pueden utilizar el sistema (*Users.cfg*).

Este fichero se encuentra en la carpeta descomprimida que contiene el código del Trabajo de Fin de Grado y que se descargó en el apartado 8 de este manual.

En dicho fichero de texto, cada línea representa a un usuario autorizado. Por tanto, en cada línea del fichero habrá que poner, en primer lugar, el nombre y los apellidos del usuario, en segundo lugar, un guión como delimitador, en tercer lugar, el correo electrónico de dicho usuario con un formato correcto, en cuarto lugar, otro guión como delimitador y, en quinto lugar, el rol del usuario (*Administrator* en caso de que el usuario sea administrador y *Normal* en caso de que el usuario sea un usuario normal). Por tanto, el usuario que vaya a ser administrador del sistema debe ponerse como rol *Administrator* para poder realizar todas las funciones que tiene el sistema.

Es importante rellenar correctamente este fichero, ya sea tanto en el formato del fichero (para que el servidor pueda realizar su lectura correctamente) como en los usuarios que se ponen en este fichero (ya que no la inclusión de un usuario evitará que dicho usuario pueda utilizar el sistema).

Un formato correcto del fichero *Users.cfg* sería, por ejemplo, el siguiente:

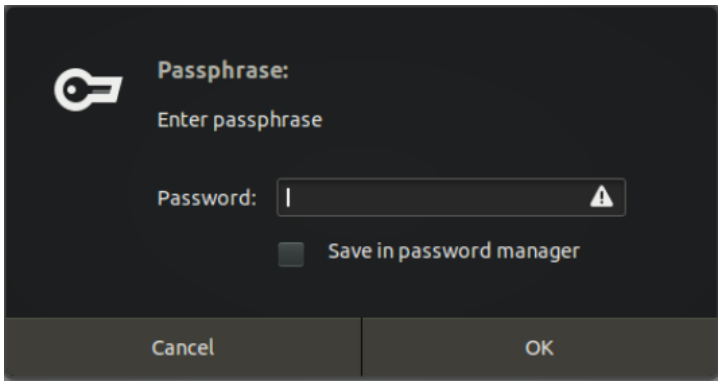


Una vez que se haya terminado de editar el fichero *Users.cfg*, habrá que cifrarlo antes de entregárselo con el resto del código del Trabajo de Fin de Grado a los usuarios normales.

Por ello, se deberá acceder por consola al directorio donde se encuentra el fichero *Users.cfg*. Una vez se esté en dicho directorio, se ejecutará el siguiente comando para cifrar el fichero:

```
> gpg -c Users.cfg
```

Tras realizarlo, aparecerá una ventana emergente que solicitará introducir dos veces la contraseña que se va a utilizar para cifrar el fichero, debiendo usar *NAOTFG2019*:



Una vez introducida la contraseña, se generará en el mismo directorio un fichero *Users.cfg.gpg*, que será el fichero de usuarios autorizados cifrado que tendrán los usuarios normales.

Si se quisiera recuperar el fichero original a partir del fichero cifrado, habría que ejecutar en la consola el siguiente comando (introduciendo posteriormente la contraseña mencionada):

```
> gpg Users.cfg.gpg
```

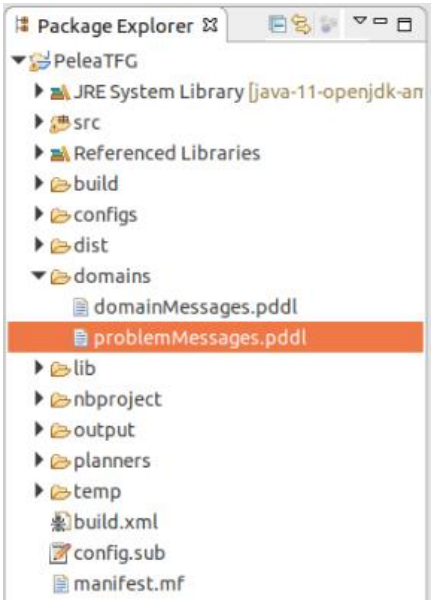
Una vez realizados ambos pasos, se generaría de nuevo el fichero *Users.cfg* original.

10. Configuración del fichero PDDL del problema

El sistema desarrollado permite entregar mensajes de un usuario emisor a un usuario receptor. Esta entrega se puede realizar haciendo que el NAO entregue un correo electrónico que contenga el audio con el mensaje grabado o haciendo que el propio NAO entregue dicho audio realizando una ruta compuesta por marcas que el NAO reconoce, como se puede ver explicado en subapartado 4.6 del manual de usuario.

En caso de que se quiera entregar el mensaje haciendo que el propio NAO sea quién lo lleve, antes de utilizar el sistema desarrollado será preciso modificar el fichero PDDL que contiene el problema que la Planificación Automática debe resolver (*problemMessages.pddl*), donde se debe configurar el mapa de marcas que el NAO deberá utilizar.

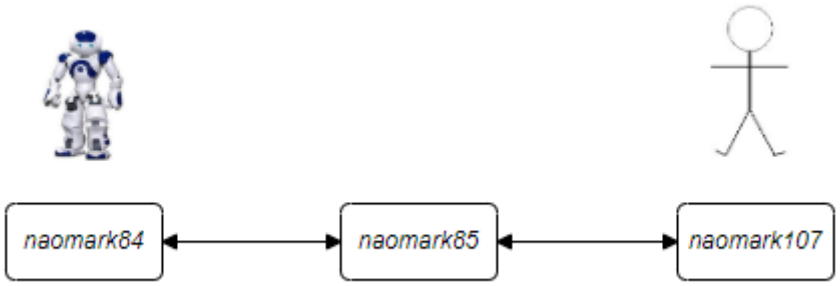
Este fichero se encuentra en la siguiente ubicación del proyecto de PELEA:



E inicialmente presenta el siguiente estado:

```
(define (problem SendingMessagesProblem)

    (:domain
        SendingMessages
    )

    (:objects
        function - function
        objective - objective
        robot0 - robot
        message0 - message
        person0 - person
        naomark84 - wayPoint
        naomark85 - wayPoint
        naomark107 - wayPoint
    )

    (:init
        (undefined function)
        (available robot0)
        (init message0)
        (origin naomark84)
        (stay robot0 naomark84)
        (connected naomark84 naomark85)
        (connected naomark85 naomark107)
        (connected naomark107 naomark85)
        (connected naomark85 naomark84)
        (stand person0 naomark107)
    )

    (:goal
        (achieved objective)
    )

)
```

Con la configuración de marcas actual, se tiene el siguiente mapa:



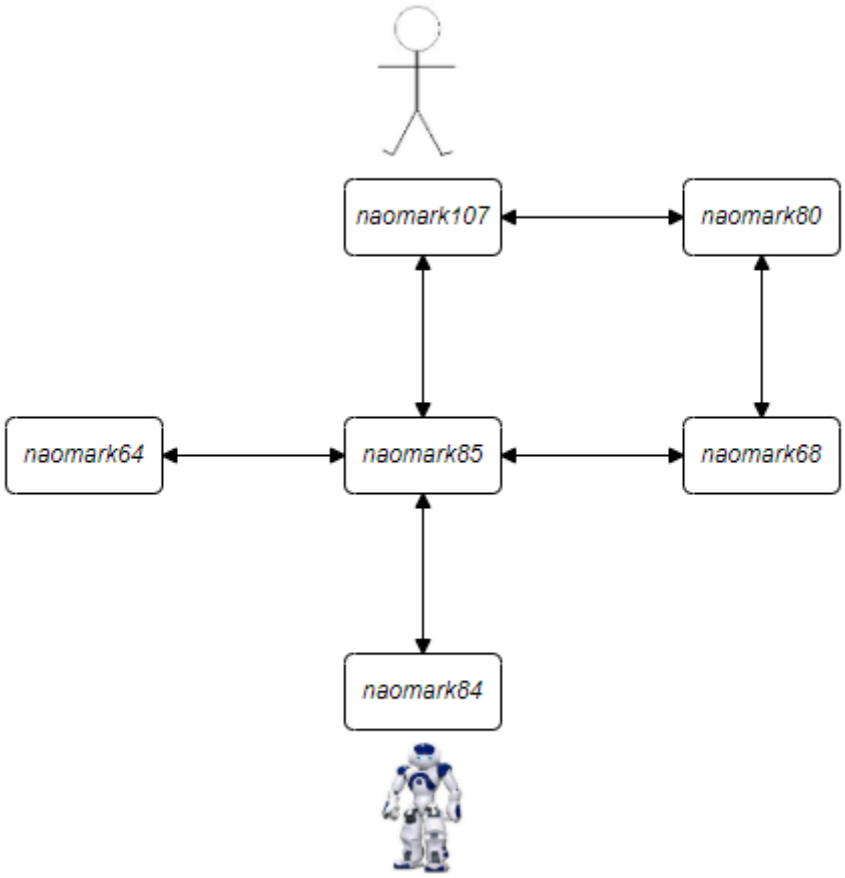
En el que el robot parte desde la marca 84, la persona está en la marca 107 y el orden de visita de las marcas en la ruta de ida debe ser 84-85-107. Una vez que el usuario receptor reciba el mensaje, podrá contestar con otro mensaje haciendo que el NAO lo lleve realizando una ruta de vuelta, cuyo recorrido de marcas deberá ser 107-85-84.

Este ejemplo inicial es muy sencillo, puesto que sólo tiene tres marcas y todas estas forman parte de la ruta que el NAO sigue, pero la complejidad del mapa podría ampliarse sin ningún problema

Por tanto, en el caso de que se quiera cambiar el fichero del problema, habrá que borrar todas las líneas del fichero relacionadas con la ruta de marcas, dejando el fichero de la siguiente forma:

```
(define (problem SendingMessagesProblem)

    (:domain
        SendingMessages
    )

    (:objects
        function - function
        objective - objective
        robot0 - robot
        message0 - message
        person0 - person
    )

    (:init
        (undefined function)
        (available robot0)
        (init message0)
    )

    (:goal
        (achieved objective)
    )

)
```

Una vez eliminadas dichas líneas, se puede plantear un problema de complejidad mayor:



En este mapa, se puede ver como la ruta más rápida que debe seguir el NAO para llegar a la persona sigue siendo desde la marca 84 hasta la 107, pasando por la 85. Pero, en este caso, se han introducido caminos que no llevan a la persona (marca 64) o caminos alternativos, pero más largos, que permiten llegar a ella (marcas 68 y 80). Para crear este problema, habría que realizar en el fichero *problemMessages.pddl* los siguientes cambios:

- Añadir en la sección de *objects* cada una de las marcas que tendrá el mapa (inicialmente en el fichero, como son tres marcas, hay tres líneas, una por cada marca). En el nuevo ejemplo habría que añadir:
  - *naomark64* - *wayPoint*.
  - *naomark68* - *wayPoint*.
  - *naomark80* - *wayPoint*.
  - *naomark84* - *wayPoint*.
  - *naomark85* - *wayPoint*.
  - *naomark107* - *wayPoint*.

Las marcas que reconoce el NAO son:

- *naomark64*.
- *naomark68*.
- *naomark80*.
- *naomark84*.
- *naomark85*.
- *naomark107*.
- *naomark108*.
- *naomark112*.
- *naomark114*.
- *naomark119*.

➢ Añadir en la sección de *init* las siguientes líneas:

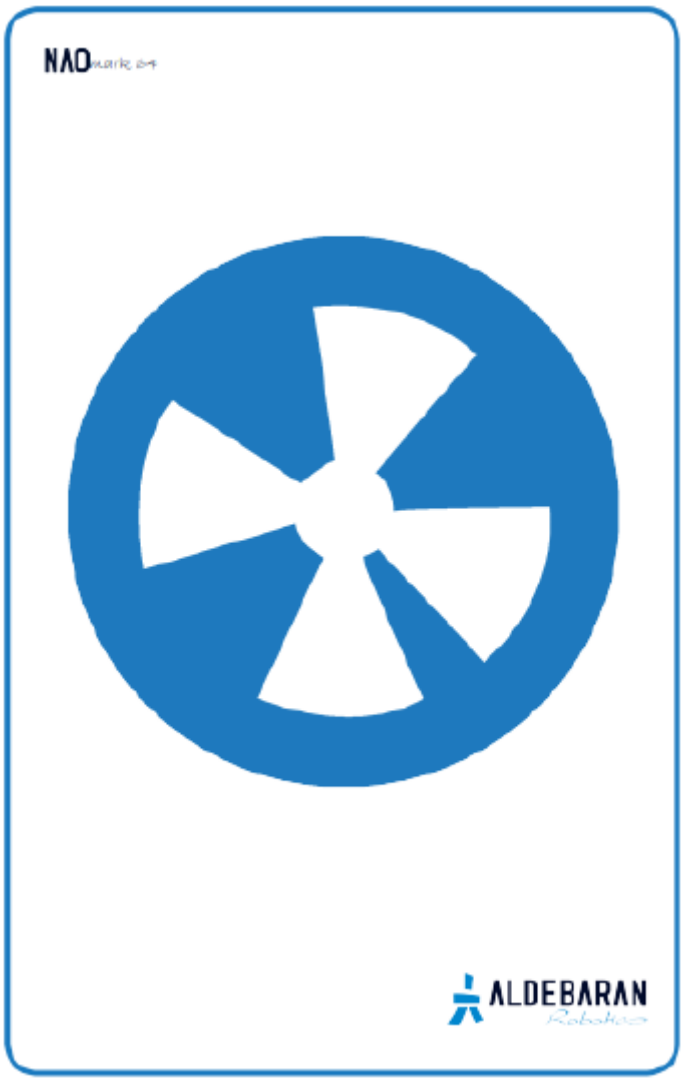
- Una línea para indicar el origen del que parte el robot:
  - *origin naomark84*.
- Una línea para indicar la posición en la que se encuentra el robot al inicio de la ruta. La marca empleada aquí debe ser la misma que la empleada en la línea anteriormente explicada:
  - *stay robot0 naomark84*.
- Las líneas necesarias para realizar las conexiones entre las marcas del mapa:
  - *connected naomark84 naomark85*.
  - *connected naomark85 naomark84*.
  - *connected naomark85 naomark64*.
  - *connected naomark64 naomark85*.
  - *connected naomark85 naomark68*.
  - *connected naomark68 naomark85*.
  - *connected naomark68 naomark80*.
  - *connected naomark80 naomark68*.
  - *connected naomark80 naomark107*.
  - *connected naomark107 naomark80*.
  - *connected naomark85 naomark107*.
  - *connected naomark107 naomark85*.
- Una línea para indicar la posición en la que se encuentra el usuario receptor:
  - *stand person0 naomark107*.

Una vez que se hayan añadido todas estas modificaciones, la configuración del fichero *problemMessages.pddl* habrá terminado.

Por último, se deben colocar físicamente las marcas que el NAO reconoce de forma que desde cada marca haya al menos otra marca visible y que cada posible receptor del mensaje esté cerca de una de las marcas.

Cada una de las marcas están disponibles en el fichero *Naomarks.pdf*, debiéndose imprimir las que se vayan a utilizar. Las impresiones que se realicen de las marcas pueden ser tanto a color como en blanco y negro y no debe haber obstáculos en la ruta entre una marca y otra.

Por ejemplo, la marca 64 es la siguiente:



Finalmente, cuando las marcas ya estén colocadas físicamente, ya se podrá mandar al NAO a que entregue un mensaje a una persona de forma física.

ANEXO C: MANUAL DE USUARIO

1. Encendido del NAO

Para encender el NAO, se deberá pulsar en el botón central situado en su pecho. Si todos los pasos explicados en el apartado 6 del manual de instalación se han realizado, al minuto aproximadamente de haber pulsado el botón, el NAO pronunciará la palabra *Ready* y estará listo para ser utilizado.

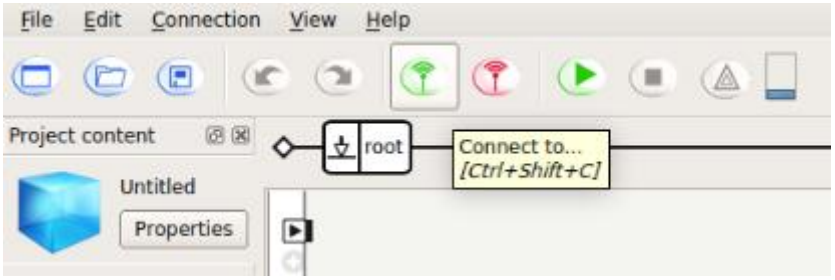
2. Arranque del servidor

Para arrancar el servidor de Python del sistema desarrollado, se deberán conocer la dirección IP y el puerto en el que el NAO está actuando, ya que son argumentos que necesita el servidor para poder arrancar.

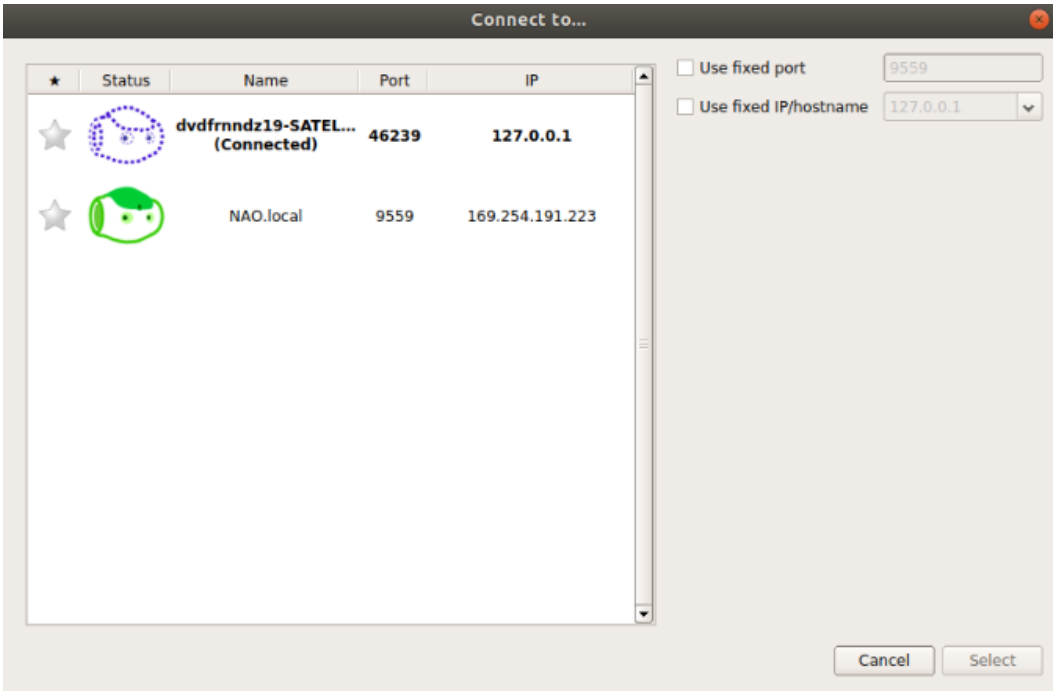
Una herramienta que facilita bastante esta labor es Choregraphe, el cual para arrancarlo habrá que iniciar la consola del sistema operativo, moverse hasta el directorio *choregraphe-suite-2.1.4.13-linux64* y realizar el siguiente comando:

```
> ./choregraphe
```

Una vez que Choregraphe esté arrancado, se deberá pulsar el botón seleccionado en la siguiente imagen:



Una vez se pulse el botón, aparecerá la siguiente ventana:



En ella, se verá con un tono verde el NAO que se va a utilizar y se podrá ver el puerto y la dirección IP que está utilizando.

Cuando ya se conozcan la dirección IP y el puerto que utiliza el NAO, se usará de nuevo la consola del sistema operativo para moverse hasta el directorio donde se encuentra el fichero *Server.pyc*.

Una vez que se esté en dicho directorio, se deberá realizar el siguiente comando:

```
> python2.7 Server.pyc --ip --port
```

Donde los argumentos son:

- --ip: dirección IP del NAO.
- --port: puerto del NAO.

Dado que el servidor lee los usuarios autorizados del fichero *Users.cfg.gpg*, el fichero Python del servidor y el fichero de usuarios autorizados deberán estar en el mismo directorio.

En caso de que se quiera detener el servidor, habrá que pulsar *Ctrl+C*.

3. Arranque de PELEA

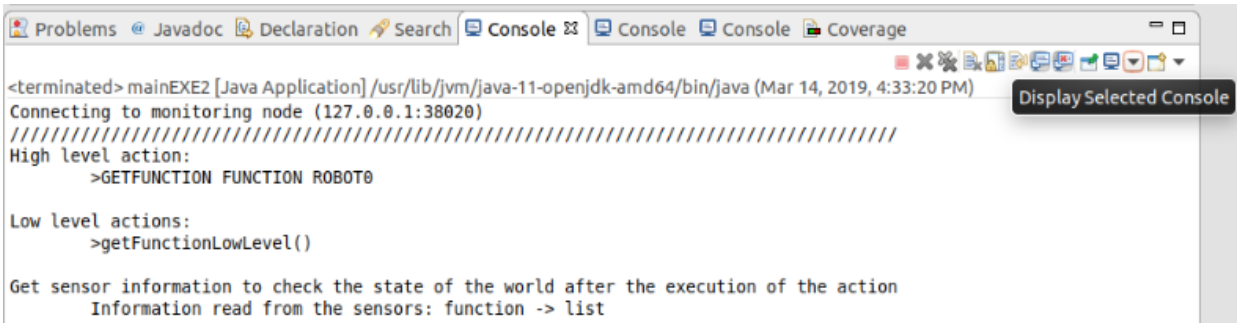
Una vez que el servidor haya sido arrancado, habrá que abrir Eclipse. Una vez abierto, se deberán seguir los siguientes pasos:

- Pulsar *Run* y *Run Configurations*.
- En la ventana que se muestre, se deberá ir a la sección de *Java Aplication* y pulsar sobre una de las clases principales configuradas (*main.java*, *mainDS.java* o *mainEXE.java*) y, una vez seleccionada, se deberá pulsar sobre *Run*.

Se deberán seguir estos pasos para cada una de las tres clases principales del proyecto (*main.java*, *mainDS.java* y *mainEXE.java*, en este orden en específico). Una vez realizado esto, el sistema desarrollado comenzará su ejecución.

Una vez que el NAO haya terminado de realizar una de las funciones que permite realizar el sistema desarrollado, habrá que realizar de nuevo los pasos de este apartado para poder realizar con el NAO una nueva función.

Por último, en caso de que se quieran observar las trazas de los módulos *Monitoring*, *Decision Support* y *Execution* de PELEA, se puede ir cambiando la vista de la consola de Eclipse pulsando el botón seleccionado en la siguiente imagen:



4. Utilización del sistema

Una vez que se hayan realizado todos los pasos de los apartados anteriores, el sistema desarrollado iniciará su ejecución haciendo que el NAO salude al usuario y le pida que le muestre su rostro. Una vez que el NAO evalúe el rostro y el usuario sea un usuario autorizado, el NAO le dirá al usuario el rol que tiene asignado en el sistema y, en función de dicho rol, le describirá las funciones que tiene disponibles. Si el usuario tiene rol de administrador, podrá realizar todas las funciones del sistema y si tiene rol normal, sólo podrá realizar las funciones de comunicarse y de salir.

4.1 Opción de guardar rostro

Esta opción permite que el usuario administrador almacene dentro del NAO el rostro de un usuario. Al iniciarse esta opción, el NAO pedirá el nombre y los apellidos del rostro que se quiere guardar. Una vez que el usuario administrador le haya dado dicha información, el NAO pedirá que se ponga delante de su cara el rostro a guardar. Cuando el rostro se haya puesto delante de la cara de NAO, será guardado por el sistema desarrollado y la opción habrá finalizado.

Esta opción no permitirá guardar un rostro en los siguientes casos:

- El usuario del que se quiere guardar el rostro no está autorizado.
- El rostro ya está guardado dentro del NAO.

4.2 Opción de buscar rostro

Esta opción permite que el usuario administrador conozca si el rostro de un usuario ya está guardado dentro del NAO o no. Al iniciarse esta opción, el NAO pedirá el nombre y los apellidos del rostro que se quiere buscar. Una vez que el usuario administrador le haya dado dicha información, el NAO informará al usuario administrador si el rostro ya está guardado o no.

Esta opción no permitirá buscar un rostro si no hay rostros almacenados dentro del NAO.

4.3 Opción de listar todos los rostros

Esta opción permite que el usuario administrador pueda ver en una consola todos los rostros que están guardados dentro del NAO. Al iniciarse esta opción, el sistema desarrollado mostrará en la consola donde se arrancó el servidor la lista de rostros almacenados dentro del NAO.

Esta opción no permitirá listar todos los rostros guardados si no hay rostros almacenados dentro del NAO.

4.4 Opción de eliminar rostro

Esta opción permite que el usuario administrador elimine del NAO el rostro de un usuario. Al iniciarse esta opción, el NAO pedirá el nombre y los apellidos del rostro que se quiere eliminar. Una vez que el usuario haya dado dicha información, el sistema desarrollado eliminará el rostro y la opción habrá finalizado.

Esta opción no permitirá eliminar un rostro en los siguientes casos:

- No hay rostros almacenados dentro del NAO.
- El rostro no está guardado dentro del NAO.

4.5 Opción de eliminar todos los rostros

Esta opción permite que el usuario administrador pueda eliminar todos los rostros que están guardados dentro del NAO. Al iniciarse esta opción, el sistema desarrollado eliminará todos los rostros almacenados dentro del NAO.

Esta opción no permitirá eliminar todos los rostros guardados si no hay rostros almacenados dentro del NAO.

4.6 Opción de comunicarse

Esta opción permite que el usuario emisor, que es la persona que está utilizando el sistema desarrollado, mande un mensaje a un usuario receptor, que es la persona que va a recibir el mensaje.

Al iniciarse esta opción, el NAO pedirá al usuario emisor el nombre y los apellidos del usuario receptor. Una vez que el usuario emisor haya dado esta información y si el usuario receptor es un usuario autorizado, el sistema desarrollado ya tendrá todos los datos necesarios tanto del usuario emisor como del usuario receptor.

Después, el NAO informará al usuario emisor de tres posibles duraciones del mensaje, de las cuáles deberá escoger una:

- Diez segundos.
- Treinta segundos.
- Sesenta segundos.

Una vez que el usuario emisor elija la duración, el sistema desarrollado grabará durante dicha duración el mensaje que se quiera transmitir.

Cuando el mensaje ya haya sido grabado, el NAO preguntará al usuario si el mensaje que quiere mandar es personal o no lo es. Dependiendo de la respuesta, la entrega del mensaje será distinta:

- Si se contesta que el mensaje es personal, el NAO adjuntará el audio grabado a un correo que enviará al correo electrónico del usuario receptor.

- Si se contesta que el mensaje no es personal, el NAO empezará a realizar una ruta compuesta por marcas que reconoce y que le servirán de pista de por dónde debe ir, buscando la primera marca definida. Una vez que haya alcanzado la última marca de la ruta, rotará sobre sí mismo buscando personas y cuando encuentre una, comprobará si es el usuario receptor. Si la persona no es el usuario receptor, el NAO seguirá buscando personas y si la persona es el usuario receptor, reproducirá el audio que contiene el mensaje grabado. Por último, el NAO preguntará al usuario receptor si quiere contestar al mensaje. En caso de que el usuario receptor diga que no, terminará la ejecución del sistema y en caso de que diga que sí, le preguntará cuál es la duración del mensaje que quiere mandar y se repetirá todo el proceso explicado ya en este apartado, sólo que en este caso será el usuario receptor el que interactúa con el NAO en vez de ser el usuario emisor. De esta forma, el usuario emisor y el usuario receptor pueden estar mandándose mensajes mutuamente hasta que uno de los mensajes grabados se envíe por correo o no se quiera enviar una respuesta al mensaje recién recibido, donde en ambos casos se producirá el final de la ejecución del sistema.

Finalmente, en caso de que el mensaje que se quiera mandar no sea personal, se deberán tener en cuenta las siguientes consideraciones:

- El usuario receptor, además de estar autorizado, debe tener su rostro almacenado en el NAO.
- El usuario administrador deberá configurar el fichero PDDL del problema con las marcas que debe seguir el NAO en su recorrido, como se explica en el apartado 10 del manual de instalación.
- Las marcas que debe seguir el NAO deberán estar colocadas físicamente y en el orden en el que fueron configuradas en el fichero PDDL del problema, pudiendo ver desde cada marca tanto la marca previa de la que se viene como la marca siguiente a la que se va.
- La altura a la que deben estar colocadas las marcas que sigue el NAO debe ser similar a la estatura del propio NAO.
- El NAO deberá estar apoyado en el suelo para que pueda levantarse y andar en el momento en el que tenga que ir a buscar las diferentes marcas de su recorrido.

4.7 Opción de salir

Esta opción permite que el usuario pueda detener la ejecución del sistema desarrollado.